60435400

CONTROL DATA CORPORATION

NOS VERSION 1
REFERENCE MANUAL

Volume 1 of 2

CDC® COMPUTER SYSTEMS:
CYBER 170 SERIES
CYBER 70
MODELS 71, 72, 73, 74
6000 SERIES

REVISION RECORD

REVISION	DESCRIPTION
A	Manual released. This manual reflects NOS 1.0 at PSR level 404.
(06-17-75)	
B	Revised to reflect NOS 1.1 at PSR level 419. New features include support of memory increments
(03-08-76)	to 262K on CDC CYBER 170 Series Systems, 844-41 Disk Storage Subsystem, multi-
	mainframe, additional security control, the Text Editor utility, and BASIC version 3. Other
	additions include: description of reserved file names in section 2, new error messages, and new
	parameters on the BLANK, CONVERT, DAYFILE, ENQUIRE, FTN, LDI, L072, and SUMMARY
	statements. Section 4 has been reorganized to more accurately describe the system control
	language. In addition, the description of OPLEDIT usage has been removed from section 14 and is
	included in the Modify Reference Manual. The entire description of the FAMILY and SYSEDIT
	statements has been removed from section 14 and is included in the NOS Installation Handbook.
	This edition obsoletes all previous editions.
C	Revised to reflect NOS 1.2 at PSR level 439. New features include revised field length control,
(12-03-76)	added security for the CHANGE and PASSWOR control statements, queued file management,
	security count, SRU limit control, and additional parameters for the LIMITS statement. The
	parameters for the COBOL 5 statement have been added to the product set descriptions. Four new
	control statements are described: MFL, ROUTE, SETASL, and SETJSL. New examples are
	included for creating multifiles on tape and using LIBEDIT. Technical and literary corrections
	have been made.
D	Revised to reflect NOS 1.2 at PSR level 452 and to make typographical and technical corrections.
(07-15-77)	The revision includes the TCOPY control statement, extensions to the COPY and VERIFY control
	statements, and support of the CDC CYBER 171 computer system. In addition, the error messages
	in appendix B have been reformatted.
E	Revised to reflect NOS 1.2 at PSR level 460 and to make literary and technical corrections.
(11-21-77)	
F	Revised to reflect NOS 1.3 at PSR level 472. This revision adds descriptions of the following
(05-26-78)	new control statements: BEGIN, DMDECS, DMPECS, ENTER, NOTE, and PROTECT. The V
	carriage control character for programmable format is outlined. The new CDC CYBER Control
	Language is presented with extensive use of examples. Section 11, Product Set Control Statements,
	was deleted. The product set control statement formats are given in the NOS Application Pro-
	grammer's Instant. This edition obsoletes all previous editions.
G	Revised to reflect NOS 1.3 at PSR level 477 and to make literary and technical corrections.
(08-25-78)	
H	Revised to reflect NOS 1.3 at PSR level 485 and to correct literary and technical errors.
(12-22-78)	

Publication No. 60435400

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

Address comments concerning this manual to:

Control Data Corporation
Publications and Graphics Division
4201 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual.

Printed in the United States of America

REVISION RECORD (CONT'D)

REVISION	DESCRIPTION
J	Revised to reflect NOS 1.4. New features in this release include CDC CYBER 170 model 176 and
(08-10-79)	885 disk support; the FCOPY, HTIME, and TRMDEF control statements; and the 12-bit ASCII
	code set. This revision contains a new section 14, Library Maintenance and a new appendix I,
	Line Printer Carriage Control. This edition obsoletes all previous editions.

Publication No.
60435400

LIST OF EFFECTIVE PAGES

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

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
Front Cover	-	1-4-10	J	1-6-22	J	1-7-47	J	1-12-7	J
Control Statements	-	1-4-11	J	1-6-23	J	1-7-48	J	1-12-8	J
Title Page	-	1-4-12	J	1-6-24	J	1-7-49	J	1-12-9	J
ii	J	1-4-13	J	1-6-25	J	1-7-50	J	1-12-10	J
ii-a/ii-b	J	1-4-14	F	1-6-26	J	1-8-1	H	1-12-11	J
iii	J	1-4-15	F	1-6-27	J	1-8-2	J	1-12-12	J
iv	J	1-4-16	F	1-6-28	J	1-8-3	J	1-12-13	J
v	J	1-4-17	F	1-6-29	J	1-8-4	J	1-12-14	J
vi	J	1-4-18	J	1-6-30	J	1-8-5	J	1-12-15	J
vii	J	1-4-19	F	1-6-31	J	1-8-6	J	1-12-16	J
viii	J	1-4-20	J	1-6-32	J	1-8-7	J	1-13-1	J
ix	J	1-4-21	J	1-6-33	J	1-8-8	J	1-13-2	F
x	J	1-4-22	J	1-6-34	J	1-8-9	J	1-13-3	F
xi	J	1-4-23	J	1-6-35	J	1-8-10	J	1-13-4	F
xii	J	1-4-24	F	1-7-1	J	1-8-11	J	1-13-5	F
xiii	J	1-4-25	J	1-7-2	J	1-8-12	J	1-13-6	J
xiv	J	1-4-26	J	1-7-3	J	1-8-13	J	1-13-7	J
xv	J	1-4-27	J	1-7-4	J	1-8-14	J	1-13-8	J
xvi	J	1-4-28	J	1-7-5	J	1-8-15	J	1-13-9	F
1-1-1	J	1-4-29	J	1-7-6	J	1-8-16	J	1-13-10	F
1-1-2	J	1-4-30	J	1-7-7	J	1-8-17	J	1-13-11	J
1-1-3	J	1-4-31	J	1-7-8	J	1-8-18	J	1-13-12	F
1-1-4	J	1-4-32	F	1-7-9	J	1-8-19	J	1-13-13	J
1-1-5	J	1-4-33	F	1-7-10	J	1-8-20	J	1-14-1	J
1-1-6	J	1-4-34	F	1-7-11	J	1-9-1	J	1-14-2	J
1-2-1	J	1-4-35	J	1-7-12	J	1-9-2	J	1-14-3	J
1-2-2	J	1-4-36	F	1-7-13	J	1-9-3	J	1-14-4	J
1-2-3	J	1-4-37	F	1-7-14	J	1-9-4	J	1-14-5	J
1-2-4	J	1-4-38	J	1-7-15	J	1-9-5	F	1-14-6	J
1-2-5	J	1-4-39	J	1-7-16	J	1-9-6	F	1-14-7	J
1-2-6	J	1-4-40	J	1-7-17	J	1-10-1	J	1-14-8	J
1-2-7	J	1-5-1	J	1-7-18	J	1-10-2	J	1-14-9	J
1-2-8	J	1-5-2	J	1-7-19	J	1-10-3	J	1-14-10	J
1-2-9	J	1-5-3	J	1-7-20	J	1-10-4	J	1-14-11	J
1-2-10	J	1-5-4	J	1-7-21	J	1-10-5	J	1-14-12	J
1-2-11	J	1-5-5	J	1-7-22	J	1-10-6	J	1-14-13	J
1-2-12	J	1-5-6	J	1-7-23	J	1-10-7	J	1-14-14	J
1-2-13	J	1-5-7	J	1-7-24	J	1-10-8	J	1-14-15	J
1-2-14	J	1-5-8	J	1-7-25	J	1-10-9	J	1-14-16	J
1-3-1	J	1-6-1	J	1-7-26	J	1-10-10	J	1-14-17	J
1-3-2	J	1-6-2	J	1-7-27	J	1-10-11	J	1-14-18	J
1-3-3	J	1-6-3	J	1-7-28	J	1-10-12	J	1-14-19	J
1-3-4	J	1-6-4	C	1-7-29	J	1-10-13	J	1-14-20	J
1-3-5	J	1-6-5	J	1-7-30	J	1-10-14	J	1-14-21	J
1-3-6	J	1-6-6	J	1-7-31	J	1-10-15	J	1-14-22	J
1-3-7	J	1-6-7	J	1-7-32	J	1-10-16	J	1-14-23	J
1-3-8	J	1-6-8	J	1-7-33	J	1-10-17	J	1-14-24	J
1-3-9	J	1-6-9	J	1-7-34	J	1-10-18	J	1-14-25	J
1-3-10	J	1-6-10	J	1-7-35	J	1-10-19	J	1-14-26	J
1-3-11	J	1-6-11	J	1-7-36	J	1-10-20	J	1-A-1	J
1-4-1	F	1-6-12	J	1-7-37	J	1-11-1	G	1-A-2	J
1-4-2	J	1-6-13	J	1-7-38	J	1-11-2	F	1-A-3	J
1-4-3	J	1-6-14	J	1-7-39	J	1-11-3	J	1-A-4	J
1-4-4	J	1-6-15	F	1-7-40	J	1-11-4	F	1-A-5	J
1-4-5	J	1-6-16	F	1-7-41	J	1-12-1	J	1-A-6	J
1-4-6	F	1-6-17	F	1-7-42	J	1-12-2	J	1-A-7	J
1-4-7	J	1-6-18	J	1-7-43	J	1-12-3	J	1-A-8	J
1-4-8	J	1-6-19	H	1-7-44	J	1-12-4	J	1-A-9	J
1-4-9	J	1-6-20	H	1-7-45	J	1-12-5	J	1-A-10	J
		1-6-21	J	1-7-46	J	1-12-6	J	1-A-11	J

PAGE	REV
1-A-12	J
1-B-1	J
1-B-2	J
1-B-3	J
1-B-4	J
1-B-5	J
1-B-6	J
1-B-7	J
1-B-8	J
1-B-9	J
1-B-10	J
1-B-11	J
1-B-12	J
1-B-13	J
1-B-14	J
1-B-15	J
1-B-16	J
1-B-17	J
1-B-18	J
1-B-19	J
1-B-20	J
1-B-21	J
1-B-22	J
1-B-23	J
1-B-24	J
1-B-25	J
1-B-26	J
1-B-27	J
1-B-28	J
1-B-29	J
1-B-30	J
1-B-31	J
1-B-32	J
1-B-33	J
1-B-34	J
1-B-35	J
1-B-36	J
1-B-37	J
1-B-38	J
1-B-39	J
1-B-40	J
1-B-41	J
1-B-42	J
1-B-43	J
1-B-44	J
1-B-45	J
1-B-46	J
1-B-47	J
1-B-48	J
1-B-49	J
1-B-50	J
1-B-51	J
1-B-52	J
1-B-53	J
1-B-54	J
1-B-55	J
1-B-56	J
1-B-57	J
1-B-58	J
1-B-59	J
1-B-60	J
1-B-61	J
1-C-1	J
1-C-2	J
1-C-3	J
1-C-4	J
1-C-5	J
1-C-6	J
1-D-1	J
1-D-2	G

PAGE	REV
1-D-3	J
1-D-4	J
1-E-1	J
1-E-2	J
1-E-3	J
1-F-1	J
1-F-2	J
1-F-3	J
1-F-4	J
1-F-5	J
1-F-6	J
1-F-7	J
1-G-1	J
1-G-2	A
1-G-3	E
1-G-4	A
1-G-5	J
1-G-6	E
1-G-7	C
1-G-8	A
1-G-9	A
1-G-10	A
1-G-11	A
1-G-12	J
1-G-13	J
1-H-1	J
1-H-2	J
1-H-3	J
1-H-4	J
1-H-5	J
1-H-6	J
1-H-7	J
1-H-8	J
1-H-9	J
1-H-10	J
1-H-11	J
1-I-1	J
1-I-2	J
1-I-3	J
1-I-4	J
1-I-5	J
1-I-6	J
Index-1	J
Index-2	J
Index-3	J
Index-4	J
Index-5	J
Index-6	J
Index-7	J
Index-8	J
Index-9	J
Index-10	J
Index-11	J
Index-12	J
Index-13	J
Index-14	J
Index-15	J
Index-16	J
Comment Sheet	J
Back Cover	-

PAGE	REV

PAGE	REV

PAGE	REV

PREFACE

This manual describes the Network Operating System (NOS) version 1.4. NOS controls the operation of CDC® CYBER 170 Series; CDC CYBER 70 Series, models 71, 72, 73, and 74; and CDC 6000 Series Computer Systems.

ORGANIZATION

The NOS Reference Manual is contained in two volumes to separate information useful only to the COMPASS programmer from information useful to all NOS users.

Volume 1 contains information for all NOS users. This includes a general description of the system and its handling of files and jobs, detailed descriptions of control language and control statement formats and processing, and explanations of memory dumps and other debugging aids. Appendixes contain NOS character sets, messages, and a glossary.

Volume 2 contains information of use primarily to the COMPASS programmer; however, several sections contain information for users of higher level languages. For reference, the table of contents of volume 2 follows the table of contents of this volume.

CONVENTIONS

Throughout this manual, cross-references to the NOS Reference Manual, volume 2, are in the form: refer to section (or appendix) n, volume 2. If volume 2 is not stipulated, the reference is to volume 1.

Uppercase letters within statement formats should be entered exactly as given; lowercase letters should be replaced with appropriate characters as described after the format.

Extended memory for the CDC CYBER 170 models 171, 172, 173, 174, 175, 720, 730, 750, and 760 is extended core storage (ECS). Extended memory for CDC CYBER 170 model 176 is large central memory (LCM) or large central memory extended (LCME). ECS and LCM/LCME are functionally equivalent, except as follows:

- LCM/LCME cannot link mainframes and cannot use a distributive data path (DDP).
- LCM/LCME transfer errors initiate an error exit, not a half exit. Refer to the COMPASS Reference Manual for complete information.

Model 176 supports direct LCM/LCME transfer COMPASS instructions (octal codes 014 and 015). Refer to the COMPASS Reference Manual for complete information.

In this manual, ECS refers to all forms of extended memory on the CDC CYBER 170 Series. However, the model 176 extended memory is excluded in references to ECS access through the DDP and to multimainframe ECS linkage.

AUDIENCE

This manual is written for all NOS users. Users can understand the manual contents without knowing the NOS assembler language, COMPASS. However, they should read the NOS 1 Batch User's Guide and/or the Network Products IAF User's Guide or the NOS 1 Time-Sharing User's Guide before reading this manual.

Users are urged to consult the glossary in appendix C for definitions of terms used in this manual.

RELATED PUBLICATIONS

Information on NOS system operation, the NOS product set, and time-sharing commands is given in the following listed manuals. Separate manuals describe CDC CYBER Record Manager and CDC CYBER Loader. The NOS Applications Programmer's Instant contains a list of the product control statements and their parameters. The Manual Abstracts booklet for NOS gives a short description of the contents of the following manuals. Refer to the Literature Distribution Services catalog for the latest revision levels.

Control Data Publication	Publication Number
ALGOL Version 4 Reference Manual	60496600
ALGOL Version 5 Reference Manual	60481600
APEX III Version 1 Reference Manual	76070000
APL Version 2 Reference Manual	60454000
Application Installation Handbook	76071100
APT IV Version 2 Reference Manual	17326900
BASIC Version 3 Reference Manual	19983900
CDCS Version 1 Reference Manual	60498700
COBOL Version 4 Reference Manual	60496800
COBOL Version 5 Reference Manual	60497100
Common Utilities Reference Manual	60495600
COMPASS Version 3 Reference Manual	60492600
CYBER Interactive Debug Reference Manual	60481400
CYBER Record Manager Advanced Access Methods Version 2 Reference Manual	60499300
CYBER Record Manager Basic Access Methods Version 1.5 Reference Manual	60495700
CYBER Loader Version 1 Reference Manual	60429800
CYBER 170 Computer Systems Models 720, 730, 750, 760, and 176 (Level B) Hardware Reference Manual	60456100
CYBER 170 Computer Systems Reference Manual	60420000
CYBER 70/Model 71 Computer System Reference Manual	60453300
CYBER 70/Model 72 Computer System Reference Manual	60347000
CYBER 70/Model 73 Computer System Reference Manual	60347200
CYBER 70/Model 74 Computer System Reference Manual	60347400

Control Data Publication	Publication Number
Data Base Utilities Version 1 Reference Manual	60498800
Data Catalogue 2 Reference Manual	60456710
DDL Version 1 Reference Manual	60359000
DDL Version 2 Reference Manual	60498400
FORM Version 1 Reference Manual	60496200
FORTRAN Extended Version 4 Reference Manual	60497800
FORTRAN Version 5 Reference Manual	60481300
FORTRAN 4 to 5 Conversion Aids Reference Manual	60483000
GPSS V/6000 Version 1 General Information Manual	84003900
LCGT/IGS Version 1 Reference Manual	17322800
Math Science Library Version 1 Reference Manual	60327500
Manual Abstracts	84000420
Modify Reference Manual	60450100
Modify Instant	60450200
Network Products Interactive Facility Version 1 Reference Manual	60455250
Network Products Interactive Facility Version 1 User's Guide	60455260
Network Products Network Access Method Version 1 Network Definition Language Reference Manual	60480000
Network Products Network Access Method Version 1 Reference Manual	60499500
Network Products Network Terminal User's Instant	60455270
Network Products Remote Batch Facility Version 1 Reference Manual	60499600
Network Products Stimulator Version 1 Reference Manual	60480500
Network Products Transaction Facility Version 1 CRM Data Manager Reference Manual	60456710
Network Products Transaction Facility Version 1 Data Manager Reference Manual	60455350
Network Products Transaction Facility Version 1 Reference Manual	60455340
Network Products Transaction Facility Version 1 User's Guide	60455360
Network Products 2550 Communications Control Program Version 3 Diagnostic Operator Handbook	60471500
Network Products 2550 Communications Control Program Version 3 Reference Manual	60471400
NOS Version 1 Applications Programmer's Instant	60436000
NOS Version 1 Batch User's Guide	60436300
NOS Version 1 Diagnostic Index	60455720
NOS Version 1 Export/Import Reference Manual	60436200

Control Data Publication	Publication Number
NOS Version 1 Installation Handbook	60435700
NOS Version 1 Operator's Guide	60435600
NOS Version 1 System Maintenance Reference Manual	60455380
NOS Version 1 System Programmer's Instant	60449200
NOS Version 1 Reference Manual, Volume 2	60445300
NOS Version 1 Terminal User's Instant	60435800
NOS Version 1 Time-Sharing User's Guide	60436400
NOS Version 1 Time-Sharing User's Reference Manual	60435500
On-Line Maintenance Software Reference Manual	60454200
PERT/Time Version 1 Reference Manual	60133600
PL/I Reference Manual	60388100
Query Update Version 2 Reference Manual	60384900
Query Update Version 3 Reference Manual	60498300
SIMSCRIPT Version 3 Reference Manual	60358500
SIMULA Version 1 Reference Manual	60234800
Sort/Merge Versions 4 and 1 Reference Manual	60497500
SYMPL Version 1 Reference Manual	60496400
TAF/TS Version 1 CRM Data Manager Reference Manual	60456700
TAF/TS Version 1 Data Manager Reference Manual	60453100
TAF/TS Version 1 Reference Manual	60453000
TAF/TS Version 1 User's Guide	60436500
Text Editor Reference Manual	60436100
Total Universal Version 1 Reference Manual	76070300
Update Reference Manual	60449900
XEDIT Version 3 Reference Manual	60455730
8-Bit Subroutines Version 1 Reference Manual	60495500
6400/6500/6600 Computer Systems Reference Manual	60100000

DISCLAIMER

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

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

VOLUME 2

APPENDIXES

SYSTEM DESCRIPTION 1

NOS is capable of several concurrent processing modes. The following are the available processing modes.

- Local batch.
- Remote batch.
- Transaction.
- Time-sharing.

The network processing modes (remote batch, transaction, and time-sharing) operate through the Network Access Methods (NAM) communications software. These processing modes are implemented, respectively, by the following NAM applications: Remote Batch Facility (RBF), Transaction Facility (TAF), and Interactive Facility (IAF).

NOS can also perform time-sharing and transaction processing through the time-sharing executive and remote batch processing through Export/Import.

The primary emphasis of this manual is on local batch processing. Users of the other processing modes should consult the appropriate manual listed in the preface.

NOS, like all operating systems, is the interface between user software and the capabilities of system hardware components. The remainder of this section describes the hardware and software that make up a NOS-controlled computer system. In most cases, the user of this manual need not understand the operation of system hardware or the internal operation of system software. This manual describes these topics only as general background for understanding NOS control statements.

SYSTEM HARDWARE

NOS can operate within the CDC CYBER 170 Series; CDC CYBER 70 Series, Models 71, 72, 73, and 74; and 6000 Series Computer Systems. The primary hardware components of each system are the following.

- Central processor unit.
- Central memory.
- Extended memory.
- Peripheral processors.
- Peripheral equipment.

CENTRAL PROCESSOR UNIT

The central processor unit (CPU) executes instructions and manipulates and stores data retrieved from central memory. The number of CPUs within a mainframe and their type varies with the machine model. As a result, some models can execute additional COMPASS assembler instructions (refer to the COMPASS Reference Manual). These model differences do not affect applications written in higher level languages.

CDC CYBER 170 and CDC CYBER 70 Series Computer Systems have the central exchange jump/monitor exchange jump (CEJ/MEJ) feature. This feature enables the system to switch CPU control between the system monitor and other programs. The information transferred from the CPU to central memory by an exchange jump operation is called an exchange package. Section 12 describes the format and use of an exchange package dump.

CENTRAL MEMORY

The primary functions of central memory (CM) are:

- To buffer data to and from the peripheral processors.
- To transfer instructions and data to and from the CPU.

Control Points

Several jobs can reside in CM simultaneously. To separate and control each job while it is resident in CM, the system assigns it to a control point. The control point is assigned a starting CM address (its reference address or RA) and allocated an initial field length (the CM words in which the job is executed). The field length is adjusted during job execution as described in section 3. Figure 1-1-1 shows a job field length within CM.

A reference to an address outside the control point address range causes a hardware error condition and job termination.

NOS supports up to 23 simultaneous control points. The maximum field length depends on the CM size and installation parameters used to control memory usage. The system assigns the CPU to control points requiring CPU activity. Rapid switching of CPU control between control points enables jobs to execute concurrently. The exact amount of time allowed for each control point depends on system activity and system parameter settings. Thus, the time required to complete a job may vary, although the actual CPU execution time is the same.

When a job completes, aborts, or rolls out (that is, its execution is suspended), the control point is released and made available to another job.

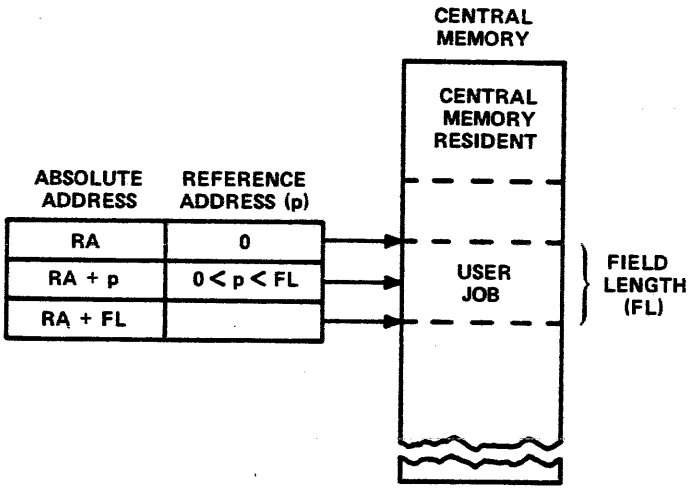


Figure 1-1-1. Central Memory Allocation

Central Memory Resident

The portion of CM reserved for system use is called central memory resident (CMR). It contains system tables, directories, and the CM portion of the system monitor (CPUMTR). Because its RA is always address 0 and its field length (FL) is the size of central memory, CMR can access any CM address and therefore specify addresses for CPU exchange jumps that switch CPU control between control points.

EXTENDED MEMORY

Extended memory (EM) is available as large central memory (LCM) or large central memory extended (LCME) on the CYBER 170 model 176 or as extended core storage (ECS) on other NOS systems. References to ECS in this manual refer to ECS, LCM, and LCME unless otherwise noted.

Slower than CM, but faster than mass storage, EM can be used for the following purposes.

- As a directly accessible memory device via FORTRAN or COMPASS statements for ECS data storage and retrieval. (Refer to the FORTRAN Extended 4 Reference Manual, FORTRAN 5 Reference Manual, or appendix D of volume 2.)
- As storage for frequently accessed data. Refer to the ASSIGN statement in section 7 and Permanent File Control Statements in section 8.
- As an alternate system device for storing copies of frequently used routines.
- As a link between mainframes in a multimainframe configuration.†

Only validated users can use EM (refer to the LIMITS statement in section 6).

† CYBER 170 model 176 extended memory cannot link mainframes.

PERIPHERAL PROCESSORS

The peripheral processors (PPs) process communications between CM and individual peripheral devices. They also perform those system control functions that are better handled by a PP than by the central processor. A peripheral processor can:

- Read and write CM.
- Read and write ECS indirectly via CM or directly via the distributive data path (DDP).†
- Transfer data to and from peripheral devices through the data channels.

NOS supports the 7, 8, 9, 10, and 20 PP configurations for 6000 series computers and 10, 14, 17, and 20 PP configurations for CDC CYBER 70 models 71, 72, 73, and 74. NOS also supports 10, 14, 17, and 20 PP configurations for all CDC CYBER 170 models except model 176. CYBER 170 model 176 has two types of peripheral processors, PPs and PPUs. The configuration supported by NOS can have 2, 4, or 6 PPUs and from 10 to 20 PPs.

For further information on PPs, refer to the appropriate system hardware reference manual listed in the preface.

PERIPHERAL EQUIPMENT

Peripheral equipment varies among installations but usually includes card readers and punches, line printers, mass storage devices, and magnetic tape units. NOS supports the following equipment models.

405 Card Reader

415 Card Punch

580-12, 580-16, and 580-20 Line Printers

844-21 Disk Storage Subsystem

844-41 and 844-44 Disk Storage Subsystems

885 Disk Storage Subsystem

667, 669, 677, and 679 Magnetic Tape Units

6671 Multiplexers for communication with 200 User Terminals and 731-12, 732-12, and 734 Remote Batch Terminals

6671 or 6676 Multiplexers for communication with interactive terminals

255x Network Processing Units

† This function does not apply to CYBER 170 model 176 peripheral processors.

SYSTEM SOFTWARE

Software executed within a computer system can be divided between software that is built into the system during system initialization and software that executes as jobs within the running system. Software present when the system begins running includes the operating system and products such as compilers, CDC CYBER Loader, and CDC CYBER Record Manager. Jobs run within the system are categorized according to their origin as described in section 3. User jobs usually consist of user programs and the system instructions required for program execution.

USER PROGRAMS

A user program is a group of CPU instructions defined by a user to perform a certain task or calculate a result. A user program can be written in a language at any of three levels.

- Compiler languages provide the user with a language suited to his particular needs. The program statements are translated by the appropriate compiler (FORTRAN, COBOL, ALGOL, and so on) that generates assembler language or machine language instructions. Programs written in compiler languages are usually machine-independent.
- Assembler languages provide a one-to-one relationship between instructions and machine operation. Mnemonics are provided for each instruction. These languages are normally used by advanced programmers because they are machine-dependent. Most of the NOS system is written in COMPASS, the assembler language of the CDC CYBER 170, CDC CYBER 70, and 6000 series computers.
- Hardware instructions are interpreted directly by the computer, and therefore, require no interpretation by a compiler or assembler. Each hardware instruction is a binary number. The programmer is rarely concerned with instructions written at this level. The exception is when program debugging requires that the user interpret memory dumps.

OPERATING SYSTEM

NOS is a group of CPU and PP programs that supervise and coordinate the operation of system hardware and the execution of products and user programs. The following lists some of the functions of NOS.

- Job validation and accounting.
- Control statement translation.
- File retrieval, manipulation, routing, and storage.
- Job input and output.
- Normal and abnormal job termination.
- Memory dumps.

CYBER Loader

CYBER Loader prepares programs for execution. Following user directions, it allocates memory for a program, loads the program modules into their appropriate locations, generates a load map, and initiates program execution. It can load subdivided programs for more efficient use of memory. Refer to the CYBER Loader Reference Manual for more information.

CYBER Record Manager

CYBER Record Manager (CRM) is the interface between user I/O functions and NOS physical I/O functions. Some of the products that use CRM are COBOL 4, COBOL 5, FORTRAN Extended 4, FORTRAN 5, Sort/Merge 4, ALGOL 4, ALGOL 5, PL/I, and DMS-170.

The functions of CRM are divided between two processors, Basic Access Methods (BAM) and Advanced Access Methods (AAM). BAM handles sequential and word-addressable file organizations; AAM handles indexed sequential, direct access, and actual key file organizations. Refer to the appropriate CYBER Record Manager manual listed in the preface.

FILES 2

A file is the largest collection of information addressable by name. All NOS data processing involves operations performed on files. Files can be differentiated by their name, structure, or file type or by whether they are assigned to a job (NOS jobs are described in section 3).

FILE NAMES

Each file has a unique 1 to 7 alphanumeric character name.†

Examples:

A 123 TAPE 1A2B COMPILE

Several file names are reserved for system use or have special significance to the system. The following file names are reserved for use by system routines.

SCR SCR1 SCR2 SCR3 SCR4

Improper use of these file names produces the following dayfile message.

RESERVED FILE NAME.

Many NOS products such as COMPASS, FORTRAN Extended 4, and UPDATE use internal scratch files. Many of these scratch files have names beginning with ZZ. The user should avoid using the name of a product scratch file for one of his own files.

The following file names are significant because they are associated with system input, print, or punch queues or with time-sharing terminals.

INPUT OUTPUT PUNCH PUNCHB P8

Refer to the description of input, print, and punch file types for more information.

FILE STRUCTURE

File structure within a computer system has several meanings. Logical file structure is how the user orders his data. He can define this logical file structure using higher-level language statements within a source program. CRM translates the higher-level language statements into the file structure that it superimposes on the data. NOS converts the NOS file and record marks that structure a file while it is being processed within the system to their physical tape, disk, or card equivalents when the file is stored.

† Some products such as FORTRAN Extended 4, FORTRAN 5, and COBOL 5 do not support file names that begin with a digit. Refer to the product reference manual for details.

CYBER RECORD MANAGER FILE STRUCTURE

CYBER Record Manager handles input/output (I/O) for several products (refer to section 1) including FORTRAN Extended 4, FORTRAN 5, and COBOL 5. CRM superimposes its file structure on the NOS file structure. Through CRM, the user can specify a file organization, a blocking type, and a record type for his data. The file organization determines how records are accessed, the blocking type determines how CRM records are grouped on their storage media, and the record type defines the smallest unit of data CRM can retrieve. The user who wants to use CRM file functions directly should consult the CRM manuals listed in the preface.

NOS FILE STRUCTURE

A NOS file can contain more than one logical file; if it does, it is called a multifile file. A multifile file begins at beginning-of-information (BOI) and ends at end-of-information (EOI). A file within a multifile file begins either at BOI or after the end-of-file (EOF) of the preceding file. It ends at its EOF.

Each file consists of one or more records of information. A record is one or more 60-bit CM words. A record begins at the BOI, after an EOF, or after the end-of-record (EOR) of the preceding record. It ends at its EOR. The following is the structure of a single-record file.

(BOI) data (EOR) (EOF) (EOI)

The following is the structure of a multirecord, multifile file.

(BOI) data (EOR) data (EOR) (EOF) data (EOR) data (EOR) (EOF) (EOI)

The last EOF in a file is optional.

PHYSICAL FILE STRUCTURE

When NOS stores a file, it converts it to a structure that conforms to the physical characteristics of the storage medium. Its file and record marks are converted to physical BOI, EOR, EOF, and EOI indicators.

The basis of all physical file structures is the physical record unit (PRU), the amount of data that can be read or written in a single device access. Table 1-2-1 lists the PRU size, and record and file mark indicators for each supported storage device.

TABLE 1-2-1. PHYSICAL FILE STRUCTURE ON STORAGE DEVICES

Storage Device		PRU Size	Record and File Mark Indicators			
			BOI	EOR	EOF	EOI
Magnetic disk or extended memory		64 CM words. 640 60-bit chars	Disk address for the file in the NOS file name table (FNT/FST).	PRU of less than 64 words with a link to the next PRU.	Zero-length PRU (no data) with special link to next PRU.	Zero-length PRU with no forward link.
Card decks†		One card.	First card in the deck.	Card with a 7/8/9 punch in column 1.	Card with 6/7/9 punch in column 1. ††	Card with 6/7/8/9 punch in column 1.
Magnetic tape†††	I (Internal)	Integral number of CM words (0 to 512); each PRU includes a 48-bit terminator.	If unlabeled, tape mark following HDR1 label. If unlabeled, load point.	A PRU of less than 512 words with level number of 0.	Zero-length PRU whose terminator contains a level number of 17_8.	Tape mark followed by an EOF1 label.
	SI (System internal)	Integral number of CM words (0 to 512); each PRU of less than 512 words has a 48-bit terminator.	If labeled, tape mark following HDR1 label. If unlabeled, load point.	A PRU of less than 512 words with level number between 0 and 16_8.	Zero-length PRU whose terminator contains a level number of 17_8.	Tape mark followed by an EOF1 label.
	S (Stranger)	Maximum of 512 words (refer to BS parameter on COPY statement in section 7 and to appendix J in volume 2).	If labeled, tape mark following HDR1 label. If unlabeled, load point.	End of each PRU.	Tape mark.	If labeled, a tape mark followed by an EOF1 label. If unlabeled, there is no EOI indicator.
	L (Long block stranger)	No maximum defined (refer to BS parameter on COPY statement in section 7 and to appendix J in volume 2).	If labeled, tape mark following HDR1 label. If unlabeled, load point.	End of each PRU.	Tape mark.	If labeled, a tape mark followed by an EOF1 label. If unlabeled, there is no EOI indicator.
	F (Foreign)	Determined by C or FC parameter on ASSIGN, LABEL, or REQUEST statement.	Load point.	None.	Tape mark.	None.

† For more information, refer to appendix F.
†† The EOF card is not recognized in a remote batch job.
††† For more information, refer to section 10 and appendix G.

Card Files

The physical file and record marks of a card file are shown in figure 1-2-1 and listed in table 1-2-1. Although card decks do not have a defined PRU size, a card is the minimum data unit. NOS can read and punch cards in coded (Hollerith), binary, and absolute binary formats as described in appendix F. Coded cards are punched in O26 or O29 keypunch mode. The system uses the installation default keypunch mode (chosen by the installation) unless a 26 or 29 is punched in columns 79 and 80 of a job, EOR, or EOF card indicating that the subsequent cards are punched in that mode.† NOS can punch up to 80 characters on a coded card and up to 150 characters (15 CM words) on a binary card.

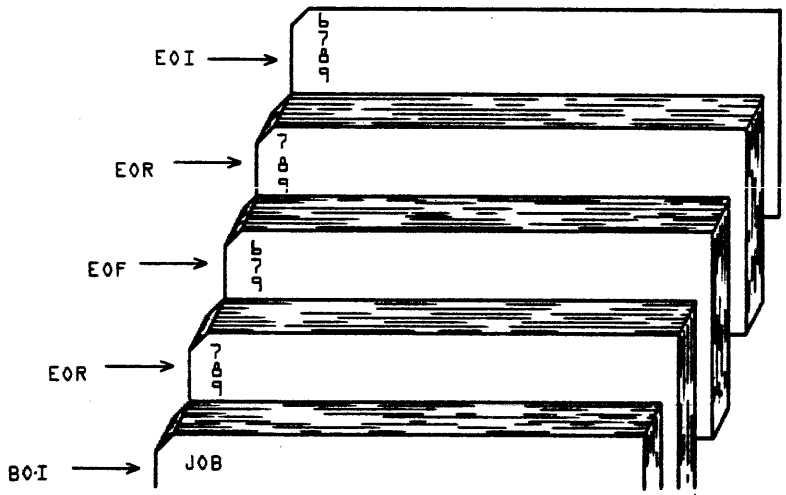


Figure 1-2-1. Sample Card File Structure

Mass Storage Files

Mass storage files are stored on disk or ECS.

The physical structure of mass storage does not concern most users; they interact with the logical structure, with logical devices and logical tracks. A logical device is one or more physical disk units known to the system as a single device. A logical track is a file allocation unit determined by the device type (refer to table 1-2-2).

† Keypunch mode selection is not supported for jobs entered through a 200UT or similar remote batch terminal.

TABLE 1-2-2. LOGICAL STRUCTURE OF SUPPORTED MASS STORAGE DEVICES

Mass Storage Device	Number of Units in a Logical Device (n)	PRUs in a Logical Track
844-21 disk (half-track)†	1 through 8	n * 107
844-21 disk (full-track)†	1 through 8	n * 112
844-41/44 disk (half- or full-track)	1 through 8	n * 227
885 disk (half- or full-track)	1 through 3	n * 640
ECS	Undefined	16

Each permanent file on mass storage is accessed via a catalog track containing the permanent file catalog of its owner. Indirect access files (refer to Permanent Files) must reside on the same device as their catalog; direct access files may reside on another device. Space is allocated for mass storage files in units called reservation blocks. An indirect access file reservation block is always 64 words (one PRU). A direct access file reservation block is a logical track. Within the user validation limits (refer to the LIMITS statement in section 6), the maximum size of an indirect access file is the device size minus space allocated for catalogs and other files. The size of a direct access file is limited only by the device size and user validation limits.

Magnetic Tape Files

NOS supports tape units that read and write 7-track and 9-track, 1/2-inch magnetic tape in binary and coded recording modes. In binary mode, NOS reads and writes 6-bit display code. In coded mode, NOS converts display code to and from coded characters. The user can select 8-bit ASCII or EBCDIC for coded 9-track tapes. Coded 7-track tapes use 6-bit external BCD code.

The user can select 200, 556, or 800 bits per inch (bpi) density for 7-track tapes or 800, 1600, or 6250 characters per inch (cpi) density for 9-track tapes, provided these densities are available with the site hardware. NOS automatically processes tape parity errors and end-of-tape conditions unless the user selects other processing options (refer to Processing Options in section 10).

† Half-track is a recording mode that accesses alternate PRUs during a disk revolution; full-track recording mode accesses consecutive PRUs. Half-track mode needs two revolutions to access all PRUs on a physical track; full-track mode needs only one revolution.

Tape Labels

Tape labels identify and delimit tape volumes and tape files. Tape marks begin and end tape labels. A tape mark is a special bit sequence written and recognized by a tape unit.

NOS processes ANSI standard and nonstandard labeled tapes. Nonstandard labeled tapes are those whose format or content do not conform to the ANSI standard described in appendix G. NOS skips to the first tape mark when reading a nonstandard labeled tape if the tape assignment statement specifies the LB=NS parameter (refer to section 10). All information after the first tape mark is then handled as data.

ANSI standard labels are those that conform to the American National Standard Magnetic Tape Labels for Information Interchange X3.27-1969 standard. NOS can create or verify ANSI labels if the LABEL statement assigns the tape file. Label verification ensures that the correct volume has been mounted. ANSI labels separate multifile set files and indicate if a file continues on another volume.

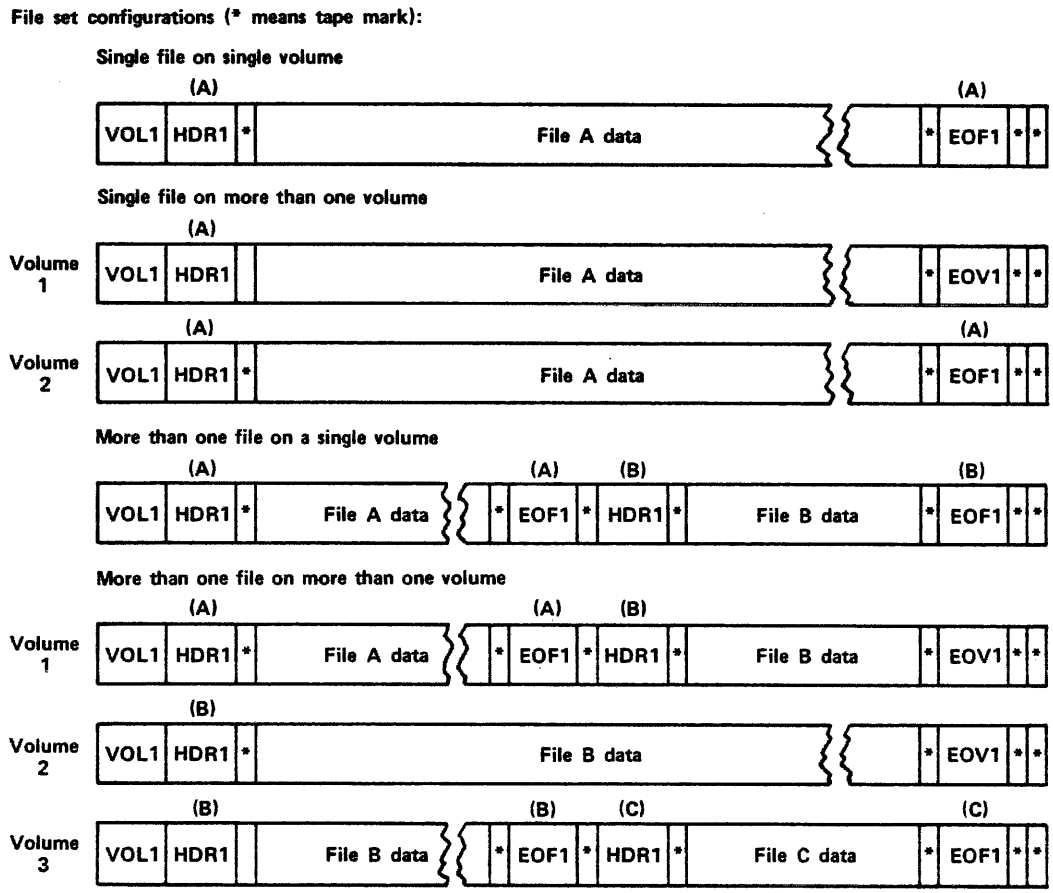


Figure 1-2-2. Use of ANSI Labels

An ANSI-labeled tape must have the following labels. Other optional labels are described in appendix G.

Label	Location
VOL1	Beginning of volume
HDR1	Beginning of information (repeated on each volume containing all or part of the file)
EOF1	End of information
EOV1	End of volume (required only if the file continues on another volume)

Appendix G gives the tape label formats. The use of ANSI labels to delimit files within file sets is illustrated in figure 1-2-2.

Tape Data Formats

NOS can read and write data on magnetic tape in any of the following formats.

Format	Mnemonic
Internal (NOS default)	I
System internal†	SI
Stranger	S
Long block stranger	L
Foreign	F

These data formats differ in their PRU (block) size and in their record and file mark indicators (refer to table 1-2-1). Other format differences are:

Tape Format	Labels	Recording Mode	Noise Size††
I	Labeled or unlabeled	Binary only	7-track: < 8 frames 9-track: < 6 frames
SI	Labeled or unlabeled	Binary only†††	7-track: < 8 frames 9-track: < 6 frames
S	Labeled or unlabeled	Binary or coded	User-selected; default is < 18 frames
L	Labeled or unlabeled	Binary or coded	User-selected; default is < 18 frames
F	Unlabeled (labels read as data)	7-track: binary or coded; 9-track: binary only	User-selected; default is < 18 frames

NOS terminates blocks on I and SI format tapes with a 48-bit block (PRU) terminator. The terminator contains the total number of bytes in the block (including the terminator itself), the number of blocks since the last HDR1 label, and the level number of the block. This terminator enables read operations on I format tapes to check if the number of bytes read and the block number expected match the byte count and block number in the terminator.

† NOS/BE system default tape format (binary mode only).
†† Tape blocks read that are smaller than the noise size are discarded. An attempt to write a block smaller than the noise size produces an error message.
††† Specification of coded mode results in job termination; refer to TCOPY statement in section 7.

If either does not match, the system attempts to recover the missing data. This feature prevents dropped or fragmented blocks and provides a higher degree of reliability than other data formats.

Tapes should be read with the same format specified as when they were written. Data is then recovered in its original form. For some formats, NOS writes extra bits which are discarded when the tape is read. I format 9-track tapes are always written with an even multiple of bytes per block. SI format 9-track tapes may have an extra 4 bits written per block to preserve the lower 4 bits of a CM word. (A 60-bit CM word would be written in eight frames, 8 bits per frame.)

All 9-track tapes are written with odd parity. Binary 7-track tapes have odd parity; coded 7-track tapes have even parity. If a parity error is detected on an F format 7-track tape, the recording mode (binary or coded) is automatically switched.

Appendix J of volume 2 describes tape formats in greater detail.

FILE TYPES

The following defines types of files assigned to user jobs and types of permanent files on mass storage. A file assigned to a job is known to the system by its entry in the file name table/file status table (FNT/FST). A FNT/FST entry contains the file name, the device on which the file resides, the file type, and its current position and status.

A permanent mass storage file is known to the system by its entry in a permanent file catalog associated with a user number. The catalog entry contains the file's name, location, length, permission modes, and access history.

FILES ASSIGNED TO USER JOBS

NOS uses the following mnemonics for file classification.

INFT	Input file	PMFT	Direct access file
PRFT	Print file	LIFT	Library file
PHFT	Punch file	ROFT	Rollout file
LOFT	Local file	TEFT	Timed/event rollout file
PTFT	Primary terminal file		

Input files, print files, punch files, rollout files, and timed/event rollout files are queued files. A queued file waits on mass storage until the system resource or peripheral equipment it requires becomes available and its priority is the highest of the files in the queue.

Input Files

An input file is also called a job file because it contains user-supplied control statements and data for a job (refer to section 3). Input files exist on mass storage in the input queue. A file enters the input queue directly when a local or remote batch job enters the system or indirectly when a user job submits another job via a SUBMIT, LDI, or ROUTE control statement. The input file of a time-sharing job consists of all terminal input directed to the system during a time-sharing session. Because the system processes the control statement immediately after it is read from the terminal, a time-sharing input file is always empty except when processing a procedure file. A user job refers to its input file by the file name INPUT (refer to Input File Control in section 3).

Print Files

A print file contains data to be printed. It is created and placed in the print queue as a result of the following.

- At job termination when the system changes the local file OUTPUT, if present, into a print file.†
- At execution of an OUT, ROUTE, or DISPOSE control statement naming a local file to be printed.

The local or remote batch subsystem processes the files in the print queue. By default, jobs originating at a central site card reader are routed to a line printer with the same ID. Similarly, remote batch output returns to the remote batch terminal where the job originated. Each remote batch terminal is given a unique terminal identification code (TID) when it logs in. Remote batch jobs and the print files they generate are given the TID of their originating terminal.

Users can override the default routing of print files with the ROUTE statement (refer to section 7). The ROUTE statement can specify a printer or printer type.

As a print file waits in the print queue, its priority increases. The file is printed when its printer becomes available and when its priority is higher than all other files destined for that printer.

OUTPUT has no special internal format. The user may wish to add appropriate printer control characters (refer to Line Printer Carriage Control in appendix I). Appendix D contains the printer output from the compilation and execution of a sample program.

Punch Files

A punch file contains data to be punched on cards. A punch file is routed from the mass storage punch queue according to the name the user assigns it or according to parameters specified on a ROUTE or DISPOSE statement. The following are punch file names.

PUNCH	Contains Hollerith punch output.
PUNCHB	Contains binary punch output.
P8	Contains 80-column absolute binary punch output.

Punch files enter the punch queue at job completion or upon execution of an OUT, ROUTE, or DISPOSE control statement. The routing and scheduling procedures for punch files are the same as for print files. Punched card formats are described in appendix F.

† Not applicable to time-sharing jobs.

Local Files

Local files are temporary files. The local file type includes all scratch and working files except the primary file.

The user can create a local file by:

- Naming the file in a COPY control statement or in a read or write statement within a program. A local file created in this manner always resides on mass storage.
- Naming the file in an ASSIGN or REQUEST control statement assigning the local file to mass storage or to a time-sharing terminal or in an ASSIGN, LABEL, or REQUEST control statement assigning the local file to magnetic tape.
- Naming the file in a GET control statement generating a local mass storage file.

To save the contents of a local mass storage file, the user issues a SAVE or REPLACE control statement to copy the local file to a permanent indirect access file. Data written on a local file assigned to magnetic tape is written on the tape for later access. Local files are released upon job completion.

Primary Files

The primary file is a temporary file designated as the primary file by a PRIMARY, NEW, or OLD control statement. Only one primary file is allowed at a time. Some control statements use the primary file as the default file when a file name is not specified. NOS rewinds the primary file before each job step.

Direct Access Files

A user assigns a direct access permanent file to his job by issuing an ATTACH or DEFINE control statement. When the user attaches the file in a mode permitting file modification, he can write on the permanent file. Refer to Permanent Files in this section.

Library Files

A library file is a read-only file that several users can access simultaneously. This file type should not be confused with system library programs or with public permanent files stored under user number LIBRARY. Refer to Libraries in this section for a description of the uses of the term library in NOS.

A user must be validated to access or create a library file. The validated user can create a library file as follows:

1. Create a local file with file name lfn.
2. Enter the following control statements.

```
LOCK(lfn)
COMMON(lfn)
```

The validated user can read a library file after naming it in a COMMON control statement.

A library file cannot be removed from the system once it has been created except by a deadstart. Library files are not retained on initial (level 0) deadstart. They are retained on level 1 or 2 deadstart if a system checkpoint was done after their creation. They are always retained after a level 3 deadstart.

Rollout Files

If, during job processing, the system or the user determines that a job must be temporarily removed from central memory, the system writes all information concerning the job on a system-defined rollout file. The rollout file includes the contents of the CM field length and the ECS field length of the job and the job-related system information from CMR. The file is read back into CM (and ECS) when the job is again assigned to a control point (refer to Rollout Control in section 3).

Timed/Event Rollout Files

A timed/event rollout file is similar to a rollout file in that it contains all the information concerning a job temporarily removed from central memory. However, a timed/event rollout file is rolled back into central memory only when a specified event has occurred (such as a file no longer busy) or a specified time period has elapsed.

A job may be written on a timed/event rollout file as a result of system or user action. The system uses a timed/event file if a job issues file or device requests that cannot be immediately honored. Users place their jobs on a timed/event rollout file when they use the ROLLOUT control statement to roll out their jobs for a specified time period.

PERMANENT FILES

Permanent files are retained on mass storage until their creator purges them. There are two types of permanent files, indirect access permanent files and direct access permanent files.

Each permanent file is listed in a permanent file catalog associated with a user number. Each permanent file catalog lists all permanent files created under that user number and their location on mass storage. Unless an alternate user number is specified, the system assumes all permanent file requests are made to this catalog.

User numbers (refer to Validation in section 3) that contain asterisks represent users with automatic read-only permission to files in the catalogs of other users. The user number must match the alternate user number in all characters not containing asterisks. For example, the user with user number *AB*DE* can access the catalogs of the following users.

UABCDEF
UABDDEE
MABCDE1

Indirect Access Permanent Files

Indirect access permanent files are accessed by copying the permanent file to a temporary file (local or primary file type). The user creates an indirect access permanent file by naming a temporary file in a SAVE or REPLACE control statement. He can retrieve a temporary copy of an indirect access file by naming it in an OLD or GET control statement. To alter the indirect access file, he enters a REPLACE statement naming the temporary copy. The system then writes the temporary copy over the permanent copy of the indirect access file.

Mass storage for indirect access permanent files is allocated in 640-character blocks (64 CM words). Because of its small allocation block size and the disk space required to maintain a working copy, indirect files are usually relatively small files.

The maximum size of an indirect access file is determined either by the value of the FS validation parameter described in the LIMITS control statement in section 6, or if no FS restriction is imposed, by the device limitations described in Mass Storage Files in this section.

Direct Access Permanent Files

The user accesses a direct access permanent file directly, not through a temporary copy. The user creates a direct access permanent file with a DEFINE control statement. He accesses the file with an ATTACH control statement. If he wants to alter the file, he must attach it in modify, append, or write mode. Only one user at a time can attach the file in write mode; this feature is called the write interlock. Data is written directly on the permanent file.

Mass storage for direct access permanent files is allocated in large blocks; the block size depends on the mass storage device type on which the file resides (refer to Mass Storage Files in this section). Because of their large allocation block size and the write interlock feature, direct access files are often used for database files.

The maximum size of a direct access file is determined by the DS validation parameter described in the LIMITS control statement in section 6, or if no DS restriction is imposed, by the device limitations described in Mass Storage Files in this section.

MASS STORAGE FILE RESIDENCE

For most mass storage file operations, the user need not be concerned about the specific device on which his file resides. However, under certain circumstances, the user may wish to override the default device residence for local or permanent files.

With the ASSIGN control statement, any user who has the necessary validation can assign a local file to either a specific device or to a device category.

Every permanent file the user creates resides either in his family of permanent file devices or on an auxiliary device. Unless the user specifies otherwise, all permanent files are saved in his family.

FAMILY DEVICES

A family consists of 1 through 63 mass storage devices. Within a family, each user has a master device that contains his permanent file catalog, all his indirect access files, and may contain some or all of his direct access files.

Normally a system has only one family of permanent file devices. However, because families are interchangeable between NOS systems, several families may be active on one system, or a system may be part of a multimainframe system. For example, consider an installation with two systems, A and B. System A provides backup service to system B. If system A failed, its family of permanent file devices could be introduced into system B without interrupting current operations on system B.

The user identifies his family by supplying a 1- to 7-character family name. The family name is included on the USER statement in batch jobs and is entered during login in time-sharing jobs. If only one family is active or if another family has been introduced into the user's normal system, he may, but need not, supply his family name. When the family name is omitted, the system uses the system default family name. If the user's family has been introduced into another system, he must supply his family name.

If the user chooses to save his files on family devices, he has the option of either using the system default device type or specifying another type of permanent file device.

AUXILIARY DEVICES

An auxiliary device is a supplement to the mass storage provided by family devices. It is identified by a 1- to 7-character pack name. An auxiliary device is not necessarily a disk pack that can be physically removed as the pack name implies. Rather, an auxiliary device can be any mass storage device supported by the system and defined as such by the installation. Each auxiliary device is a self-contained permanent file device; all direct and indirect access files represented by the catalogs on the device reside on the device. Auxiliary devices may be defined as public or private. Anyone permitted to use auxiliary devices who supplies the appropriate pack name can create, replace, and access files on a public device. Only one user, the owner, can create and replace files on a private auxiliary device, but others may access those files as permitted by the owner.

LIBRARIES

As defined in the glossary (appendix C), the term library has several meanings. The applicable meaning for the term must be determined from its context. The following describes some NOS libraries.

USER NUMBER LIBRARY

Files stored under user number LIBRARY need not be libraries themselves. An installation saves programs or text as files under user number LIBRARY so that validated users can access them. Users access those files by specifying the file name and the alternate user number LIBRARY on their permanent file request or by issuing the LIBRARY time-sharing command (refer to the IAF Reference Manual or the Time-Sharing User's Reference Manual).

PROGRAM LIBRARIES

A program library is a collection of source deck images stored in compressed Modify or Update format. The validated user accesses these compressed source decks through MODIFY or UPDATE control statements (refer to section 13).

USER LIBRARIES

User libraries are the files named in the LIBRARY loader control statement and searched by CYBER Loader to satisfy external references within the program it is loading. They contain compiled or assembled routines. The first record of a user library is a ULIB record; the last record is an OPLD directory record (refer to the LIBGEN statement in section 14).

User libraries are generated by the user, the product, or the system. CYBER Loader first searches the user-generated libraries specified by a LIBRARY or LDSET control statement (refer to the CYBER Loader Reference Manual). CYBER Loader then searches the product set library (such as the FORTRAN Extended library) stored on the system library. Finally, CYBER Loader searches the system default user library SYSLIB, which is also on the system library.

Section 14 describes control statements that catalog and manipulate library records.

JOB FLOW AND EXECUTION 3

A job is a file of statement images.† Its first record contains control statements that specify job processing requirements. Every job begins with a job statement and a USER statement. The end of the control statement record is marked by an EOR.

Records that follow the control statement record contain program, data, or directive input for processing control statements. The user can consider the job file as two files, entered as one but processed concurrently. As each control statement requiring additional user input is processed, the system reads the next record in the input file (unless the control statement specifies otherwise). These following records must be in the same order as the control statements that use them.

For example, figure 1-3-1 illustrates a basic job deck. The first three control statements are processed by system routines that require no additional user input. The fourth control statement, FTN(GO), requests two job steps, the compilation of a FORTRAN Extended program and its execution. Because the I parameter is omitted from the statement, the system reads the next record of the input file, expecting it to be a FORTRAN source program. After successful compilation, the system executes the program, taking input data from the third record of the input file. Normal job termination occurs when the system reads the control statement record EOR (the first 7/8/9 card).

JOB INITIATION

The user initiates jobs by:

- Reading a card deck in through a local or remote batch reader.
- Logging into a time-sharing terminal.
- Entering a job via an LDI, ROUTE, or SUBMIT control statement within a job already in the system.

† A time-sharing job consists of all input entered during a time-sharing session (refer to the IAF Reference Manual or the Time-Sharing User's Reference Manual).

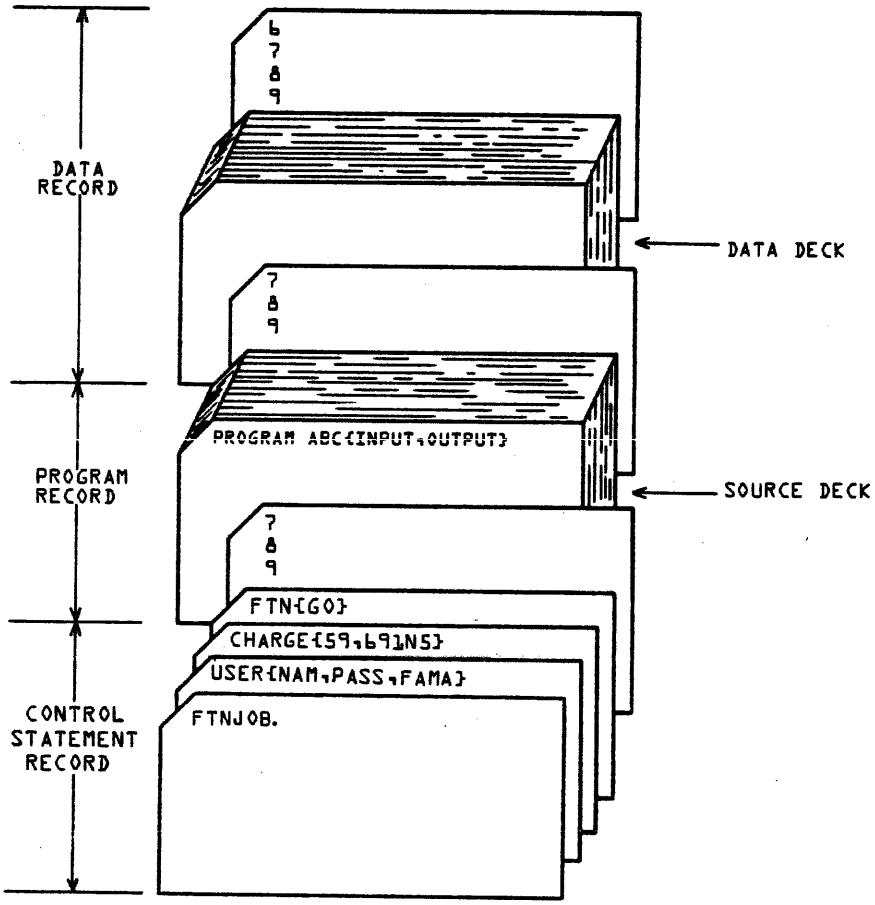


Figure 1-3-1. FORTRAN Compile and Execute Deck

JOB ORIGIN TYPES

When a job enters the system, the system determines the job origin type according to the means used for job initiation. Its origin identification remains with the job throughout job processing. The job origin type determines how the job is handled and how it exits from the system.

Jobs originating from the system console are assigned system origin type (SYOT). Jobs entered through the time-sharing executive or the Interactive Facility (IAF) are assigned time-sharing origin type (TXOT). Jobs entered through a local batch card reader are batch origin type (BCOT) jobs. Jobs entered through Export/Import or the Remote Batch Facility (RBF) are remote batch origin (EIOT) jobs.

If validated, a user can initiate jobs using the LDI, ROUTE, or SUBMIT control statements. Jobs initiated by ROUTE or SUBMIT statements can be either batch origin or remote origin jobs depending on the statement parameters. Jobs initiated by LDI statements are batch origin jobs.

JOB NAMES

After entering the system, the job is assigned a unique job name to prevent job name duplication within the system. This name is not the job name specified on the job statement. The first 7 characters of the job identification are the system-assigned job name; the eighth character indicates the job origin type. This job name precedes all messages issued to the system dayfile for that job. These messages include normal operating messages, error messages, and accounting information issued by the system.

SYSTEM ORIGIN TYPE (SYOT) JOB NAME FORMAT

The first 4 characters of a system job name are obtained from the job name entered or are zero-filled if fewer than 4 characters are entered. The next 3 characters are a unique system sequence number in the range from AAA to 999. The eighth character is an S. For example, if the job entered is DIS, a possible job name is DIS0AABS.

BATCH ORIGIN TYPE (BCOT) JOB NAME FORMAT

The first 4 characters of a batch origin job name are generated from the user index associated with the user number supplied on the USER control statement. These 4 characters are unique to the user. The next 3 characters are the job sequence number. The eighth character of a batch origin job name is B.

TIME-SHARING AND REMOTE BATCH (TXOT AND EIOT) JOB NAME FORMAT

The first 4 characters of these job names are generated from the user index associated with the user number supplied by the user when logging into the system. The next 3 characters represent the number of the terminal on which the user is logged in for TXOT or the system sequence number for EIOT. The eighth character is T for time-sharing origin jobs and E for remote batch jobs.

All jobs entered via a SUBMIT or ROUTE control statement derive the first 4 characters of their job names from the job's current user index in the same manner as EIOT and TXOT jobs. The last 3 characters are the system sequence number with the eighth character being either E or B, as described previously, depending on the parameters supplied with the SUBMIT or ROUTE statement.

VALIDATION

The USER statement follows the job statement and is used to validate the user as a legal user (refer to USER statement in section 6). If the user is validated, a set of control values is set in the control point area; these values are used by the system to control all system requests. In most cases, if the user is not permitted to perform specific functions (such as access nonallocatable devices), his job is aborted and the message

ILLEGAL USER ACCESS.

is issued when the illegal function is attempted.

To determine the extent of his validation, the user can issue the LIMITS command and receive a listing of his current validation control values. Refer to the LIMITS control statement in section 6 for an explanation of these values. For further information or to change his validation, the user should contact installation personnel.

Each user number has a unique user index associated with it. Once a user number is validated, the user index is set in the control point area. The system uses this index to determine the location of the user's permanent file catalog. (Refer to the NOS System Maintenance Reference Manual for an explanation of the user index.)

ACCOUNTING

The unit of accounting for the system is the system resource unit (SRU). The SRU is a composite value of central processor time, I/O activity, and memory usage. SRU operations are initiated at the beginning of a job and reinitiated whenever another CHARGE control statement is encountered. SRU information includes:

- Central processor time.
- Mass storage activity.
- Magnetic tape activity.
- Permanent file activity.
- SRU value.
- Application account charges.†

This information is written to the user's dayfile at the end of the job or whenever a CHARGE statement is processed. The user may request SRU information to be written to his output file at any time during the job by issuing the ENQUIRE or SUMMARY control statement. The format of SRU information written in the dayfile is given under Job Completion in this section.

†Not currently supported by the system but reserved for future use.

JOB SCHEDULING

When a job enters the system, it is placed in the input queue on mass storage, where it waits for the required system resources to become available. The job is assigned an input queue priority depending on its origin. The system priorities are system-defined and can be altered only by the system operator. The job queue priority is advanced as the job waits in the queue. The priority ages to a system-defined limit. The job scheduler periodically scans the queues and active jobs to determine whether action is necessary to ensure that the highest priority jobs are being serviced. This action may include rolling out low priority jobs or rolling in higher priority jobs. The job scheduler is also activated to analyze the system status whenver the status of the system changes (for example, when the field length of a job is released, a job enters a queue, or a job completes). Because of this automatic scheduling and analysis of system status changes, a user can increase system performance by releasing memory when all the assigned memory is not required.

Once a job is brought to a control point, normal control statement processing begins. The general flow of the control statement processing is illustrated in figure 1-5-1.

JOB CONTROL

While a job is at the control point, the system exercises the following controls over the job.

FIELD LENGTH CONTROL

The system controls the field length assigned to a job, adjusting it according to the requirements of each job step. A programmer can influence the field length assigned to his job by using the central memory job statement parameter (refer to section 5) and the MFL and RFL control statements (refer to section 6).

The maximum field length for a job (MAXFL) is set at the smallest of the following values.

- Central memory job statement parameter value, if specified.
- Maximum field length for which the user is validated.
- Maximum field length available for user jobs (dependent on machine size).

The maximum field length for each subsequent job step (MFL) is initially set equal to MAXFL. It can be reset, however, by MFL control statements. MFL cannot exceed MAXFL.

The running field length (RFL) is initially set to zero, indicating system control of field length. The RFL control statement resets RFL. RFL cannot exceed the current MFL.

To set the initial field length for a job step, the system uses the first value set by one of the following.

- Predefined initial field length for a system routine (RFL= or MFL= special entry point as described in appendix F, volume 2).
- Highest high address (HHA) from EACP loader table (54 table). Refer to the CYBER Loader Reference Manual.
- RFL value, if nonzero.

- The smaller of the MFL or the installation-defined default value (release value 50000B).

CYBER Loader further adjusts the field length during program loading. Memory may be added or removed as the needs of the program change. Refer to the description of the REDUCE control statement in the CYBER Loader Reference Manual.

The following example shows a control statement record, the MAXFL, MFL, and RFL settings, and the actual field length used to process the statement.

Control Statement	MAXFL	MFL	RFL	Field Length	Explanation
JOB(CM60000) USER(USERABC,123,FAM1) CHARGE(4922,66X)	60 000 60 000 60 000	60 000 60 000 60 000	0 0 0	700 700 2200	The CM parameter sets the MAXFL and MFL values. The system sets the field length as required for processing the control statements.
GET(ABSPROG,RELPROG)	60 000	60 000	0	1700	GET statement retrieves copies of an absolute program and a relocatable program.
RFL(40000)	60 000	60 000	0	1500	The user issues an RFL statement to set the field length for execution of the absolute program that follows.
ABSPROG.	60 000	60 000	40 000	40 000	The absolute program on file ABSPROG is executed within a 40 000-word field length.
MFL(50000)	60 000	60 000	40 000	1500	The user issues an MFL statement to set the maximum field length for the following relocatable load.
RELPROG.	60 000	50 000	0	$\leq$50 000	If more than 50 000 words is required, the job aborts.

INPUT FILE CONTROL

All user jobs, when initiated, have a file named INPUT. This file contains the control statements and other input records required for job execution. (INPUT is a locked file.) As a result, the user may read from it and reposition it, but the system does not allow him to write on it. If for some special reason the user needs to write on INPUT, he should first issue a RETURN(INPUT) control statement (refer to section 7). This statement changes the name of the file from INPUT to INPUT* and leaves it assigned to the user's job. The change of name on RETURN applies only if the input file is of type INFT.

TIME LIMIT CONTROL

The system sets a time limit for each job step unless the job statement or the SETTL statement specifies a job step time limit. This time limit is the amount of central processor time that any one job step is allowed. The user cannot increase the limit beyond that for which he is validated.

While a job is using the central processor, the time of usage is accumulated and checked against the time limit for each job step. If the job is not a time-sharing (TXOT) job, the job in execution is aborted when the time limit is reached. Time-sharing origin jobs are rolled out, after which the user can increment the time limit and resume execution from the point where the time limit was exceeded. Refer to the IAF Reference Manual or the Time-Sharing User's Reference Manual for more details.

SRU LIMIT CONTROL

The system sets a limit on the number of system resource units (SRU) that a job step or an account block can accumulate. An SRU includes central processor time, central memory usage, permanent file activity, and mass storage and tape I/O. An account block is that portion of a job from one CHARGE statement to the end of the job or the appearance of another CHARGE statement. The user may alter these limits through the SETJSL and SETASL control statements or macros; however, he may not set either limit beyond that for which he is validated.

While a job is in the system, SRU usage is accumulated and checked against the SRU step and account block limits. If the job is not a time-sharing job (TXOT), the job is aborted when either limit is reached. Time-sharing jobs are rolled out. After a time-sharing job is rolled out, the user can increment the limit and resume execution from the point where the limit was reached. Refer to the IAF Reference Manual or the Time-Sharing User's Reference Manual for more details.

CONTROL STATEMENT LIMIT CONTROL

If a job executes more control statements than the number for which the user is validated, the following message is issued when the limit is reached by job processing.

INITIAL CONTROL STATEMENT LIMIT.

The job is then allowed eight additional control statements for error processing such as saving and dumping of files. When this limit of eight statements is exceeded, the job is terminated with the following message.

CONTROL STATEMENT LIMIT.

A user's control statement limit is given by the CC field in the output from the LIMIT control statement (refer to section 6).

ROLLOUT CONTROL

Each executing program is allowed to reside in CM for a certain amount of time before relinquishing its space to another program. When this CM time slice is exceeded, the program may be rolled out. This means that the contents of the job field length, the job control area, and the control registers (exchange package) are written to mass storage. The program remains on mass storage until it is rolled back into memory. Execution resumes from the point where rollout occurred. The amount of time the job is allowed to occupy CM is called the central memory time slice. The central memory time slice is a system parameter that can be changed only by the system operator. The time slices vary for each origin type. Whether a job is rolled out when its time slice expires depends on several factors.

- Whether there are jobs waiting in the input and rollout queues.
- Whether the jobs that are waiting have a lower priority.
- Whether jobs that are waiting require more field length than would be available if all jobs of lower priority were rolled out.

When a job is rolled out, it is assigned a queue priority. The priority assigned is a system parameter and can be changed only by the system operator. The queue priorities can vary for each origin type. The queue priority is aged (incremented) while the job is in the rollout queue. Normally, all other factors being equal, the job with the highest queue priority is selected to be rolled in.

ERROR CONTROL

When job step activity ceases, the system must determine the next control statement to process. If activity ceased due to normal termination, the next control statement processed is the next statement in sequence. If an error caused activity to cease, the system issues the appropriate dayfile message and exits from the job.

Errors may be detected by system software or hardware. When the system hardware detects an error condition, NOS issues two or more dayfile messages. The first message gives the address where the error was detected. The second and following messages give the types of errors that occurred. NOS then dumps the exchange package for the job to OUTPUT (for batch origin jobs) or to the mass storage file ZZZDUMP (for time-sharing jobs) (refer to section 12).

After issuing the appropriate dayfile message(s), the system searches for an EXIT control statement. If an EXIT statement is found, processing continues with the statement following EXIT. If, before detecting the error, the system encounters a NOEXIT statement, it makes no search for an EXIT statement, and processing continues with the next control statement. If neither an EXIT nor a NOEXIT statement is encountered, the system terminates the job. (Exit processing is further described in section 5.)

The user can specify the error exit mode on which the system is to abort with the MODE statement so that address or operand out of range and/or indefinite operand errors are allowed and program execution continues (refer to section 6). The default error exit mode specifies that all errors terminate the job.

Volume 2 describes the EREXIT and MODE macros that can be used to control exit processing in COMPASS programs. Section 12 of volume 2 specifies file completion procedures when a job step abort occurs.

SECURITY CONTROL

Unless the job is system origin type or the user is validated for system origin privileges and DEBUG mode has been set at the system display console, system security imposes the following restrictions on control statements which dump any portion of the field length of the previous job step.

- They may not follow the execution of certain protected system programs (refer to section 1, volume 2, for further definition).
- They may not follow user programs which have requested protection (refer to the description of the SETSSM macro, section 6, volume 2).

If the user violates these restrictions, the system issues an informative message to the dayfile and ignores the control statement.

The following are the restricted control statements.

CATALOG	COPYCF	DMD	LBC	RESTART
CKP	COPYCR	DMDECS	LIBEDIT	TCOPY
COPY	COPYEI	DMP	LOC	VERIFY
COPYBF	COPYSBF	DMPECS	PBC	VFYLIB
COPYBR	COPYX	EDIT	RBR	WBR

CCL statements are also restricted (refer to section 4).

JOB COMPLETION

When there is no more activity at a control point, no outstanding central processor requests, and no control statements to process, the job is completed in the following manner.

1. All CM assigned to the job is returned to the system.
2. ECS assigned to the job is released.
3. All equipment assigned to the job is returned to the system.
4. All library files attached to the job are returned; other jobs can then access them.
5. All scratch (local) file space used by the job is released.
6. All direct access permanent files attached to the job are returned; the status information for these files is updated.
7. The following summations of job activity are added to the end of the user's dayfile. This information is also issued to the associated account dayfile. The entries in the account dayfile also include the job name.

   - Application charge activity in kilounits:

     hh.mm.ss.UEAD, xxxxxx.xxxKUNS.

   - Permanent file activity in kilounits:

     hh.mm.ss.UEPF, xxxxxx.xxxKUNS.

   - Mass storage activity in kilounits:

     hh.mm.ss.UEMS, xxxxxx.xxxKUNS.

   - Magnetic tape activity in kilounits:

     hh.mm.ss.UEMT, xxxxxx.xxxKUNS.

- Accumulated central processor time in seconds:†

  hh.mm.ss.UECP, xxxxxx.xxxSECS.

- SRU value in units for total job usage including CPU time, I/O activity, and memory usage:

  hh.mm.ss.AESR, xxxxxx.xxxUNTS.

- Lines printed in kilolines:

  hh.mm.ss.UCLP,mies, xxxxxx.xxxKLNS.

mi	Machine ID
es	EST ordinal of the output device

- Cards read in kilocards:

  hh.mm.ss.jobname. UCCR. mies. xxxxxx.xxxKCDS.

The following information is issued to the account dayfile only.

- Cards punched in kilocards:

  hh.mm.ss.jobname. UCPC. mies. xxxxxx.xxxKCDS.

8. Control point dayfile is copied to the end of the OUTPUT file. If an OUTPUT file does not exist or if it is a deferred routed file with EC=A9 specified, the dayfile is copied to another print file.

9. All print and punch files are released to the print and punch queues.

10. The control point area is cleared for the next job.

† If the installation defines a CPU multiplier value, the value given is the product of the actual CPU seconds and the multiplier.

INTRODUCTION

The CYBER Control Language (CCL) is a set of statements that the user can insert in the control statement record of a job to initiate tests, transfers, and loops within that record. CCL also enables the user to set values to symbolic names, display results in the dayfile of the job, and interrogate the system to determine the status of files. CCL also enables a user to create and reference procedure files which contain sequences of control statements and/or control language statements.

Another system control language is also available but users are encouraged to use CCL (refer to appendix H).

The following paragraphs describe the elements of CCL. This is followed by a description of CCL expressions. The remainder of the section describes CCL statements, functions, and procedures.

The following CCL statements are used to skip or conditionally process a sequence of statements.

Statement	Description
IFE	Passes to the next statement if an expression associated with IFE is true; skips until a terminating CCL statement is found, if it is false.
SKIP	Skips until a terminating CCL statement is found.
ELSE	Either terminates or initiates skipping, depending upon other CCL statements employed.
ENDIF	Terminates skipping initiated by IFE, SKIP, or ELSE.

The following CCL statements identify a sequence of control statements as a loop that can be repeatedly processed.

Statement	Description
WHILE	Establishes the beginning of the loop. If the associated expression is true, the loop is processed; if it is false, the loop is not processed.
ENDW	Establishes the end of the loop.

The following CCL statements assign and display values associated with symbolic names.

Statement	Description
SET	Allows the user to assign values to special CCL registers.
DISPLAY	Evaluates an expression and displays the result in the dayfile of the job.

CCL provides the following functions to be used with expressions.

Function	Description
FILE	Determines the attributes of a file.
DT	Determines the type of device on which a file resides.
NUM	Determines if a parameter has a numeric value.
SS	Determines the subsystem in use.

The following CCL statements enable the user to define and control processing of a control statement procedure.

Statement	Description
BEGIN	Initiates processing of a procedure.
REVERT	Returns processing from a procedure to the control statement record that called it.

The following CCL commands identify statements requiring special processing.

Command	Description
.PROC	Precedes and identifies a procedure.
.DATA	Precedes a sequence of statements to be written to another file.
.EOR	Separates records within the statement sequence to be written to another file.
.EOF	Separates files within the statement sequence to be written to another file.
.*	Prefixes a comment line.

All CCL statements must be terminated. Within a statement, an expression or function may end in a right parenthesis. If this occurs at the end of the statement, the right parenthesis does not serve as the statement terminator; an additional terminator must be included to complete the statement.

EXPRESSIONS

A CCL expression consists of operators and operands. Expressions can be nested within expressions by means of parentheses; however, parentheses do not imply multiplication. Operators can be arithmetic, relational, or logical. Operands can be any of the following.

Operand	Description
Integer constant	A string of 1 to 10 characters. If the string is to be a literal, it must be delimited by dollar signs ($xxxxxxxxxx$).
Symbolic name	An alphanumeric character string of 1 to 10 characters. It has a numeric value. This value is either an installation-defined constant or a user- or CCL-defined variable.
CCL function	A CCL-defined operand which determines attributes of a file or symbolic name.
Expression	A CCL expression enclosed with separators. This expression is evaluated, and the result is the operand.

An expression can be as long as the user wishes; however, there must be a period or a closing parenthesis within the first 50 operands.

Any character string beginning with a numeric character is treated as numeric. This string cannot contain any nonnumeric character except an optional postradix B (octal) or D (decimal). An alphanumeric string must begin with an alphabetic character.

CCL expressions can be used with the CCL statements IFE, WHILE, DISPLAY, and SET and with the FILE function. The separator preceding the expression can be a comma or left parenthesis. The separator following the expression must be a comma.

Integer arithmetic is used in each step of the evaluation of a CCL expression. Division, multiplication, and exponentiation produce a zero result if the absolute value exceeds $2^{48}-1$.

OPERATORS

ARITHMETIC OPERATORS

The following are the CCL arithmetic operators.

+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponentiation
Leading -	Negation
Leading +	Ignored

RELATIONAL OPERATORS

A relational operator produces a value of one if the relationship is true and zero if it is false. The following are the CCL relational operators (either form may be used).

=	.EQ.	Equal to
	.NE.	Not equal to
<	.LT.	Less than
>	.GT.	Greater than
	.LE.	Less than or equal to
	.GE.	Greater than or equal to

LOGICAL OPERATORS

When a CCL expression contains a logical operator, CCL evaluates the full 60 bits of each operand and produces a 60-bit result. If the result has any bits set, it is true; if no bit is set, the result is false. The following are the CCL logical operators.

.EQV.	Equivalence
.OR.	Inclusive OR
.AND.	AND
.XOR.	Exclusive OR
.NOT.	Complement

ORDER OF EVALUATION

The order in which operators in an expression are evaluated is:

1. Exponentiation
2. Multiplication, division
3. Addition, subtraction, negation
4. Relations
5. Complement
6. AND
7. Inclusive OR
8. Exclusive OR, equivalence

OPERANDS

INTEGER CONSTANTS

An integer constant is usually a whole number (a numeric value without a fractional component) but may be a literal (a $-delimited character string). If it is numeric, it must be 10 characters or less including an optional postradix. If no postradix is included, decimal is assumed. If an integer constant is a literal, it must be 10 characters or less, excluding the $ delimiters. If CCL encounters a literal, it is right-justified in display code and processed as a whole number.

SYMBOLIC NAMES

A symbolic name is an alphanumeric string to which a numeric value can be assigned. This numeric value may be defined at installation time or may be a variable set by the user or by CCL. All variables, except those for OT (job origin types), SYS (host operating system), VER (version of the operating system), and TIME (current time of day), have an initial value of zero.

Symbolic names with special attributes are listed in table 1-4-1.

These attributes are defined as follows:

Attribute	Description
Local	An X in this column of the table indicates that the value is saved by a BEGIN statement before initiating a procedure and restored by a REVERT statement upon termination of a procedure. (Procedures are explained later in this section.)
Set	An entry in this column specifies how the symbol obtains its value. One or more of the following characters may be listed in this column for each symbolic name.

Character	Description
B	Set by BEGIN.
O	Set by the operating system.
R	Set by REVERT.
U	Set by the user with the SET control statement or the SETJCI macro (refer to section 6 in volume 2).

Compare — The entries in this column are symbolic names with fixed values referring to errors or job status. These fixed values are compared with the symbolic names in the NAME column via a CCL statement. This produces a true or false result.

Example:

If BCO in the NAME column is equivalenced to OT in the COMPARE column with the CCL statement

```
IFE,BCO=OT,JUMP.
```

and the job is local batch, the BCO=OT expression is true and control passes to the next statement. (IFE is explained later in this section.)

TABLE 1-4-1. SYMBOLIC NAMES WITH ARITHMETIC VALUES

Name	Local	Set	Compare	Description
R1	X	U		Contents, control register 1
R2	X	U		Contents, control register 2
R3	X	U		Contents, control register 3
R1G		U		Contents, global control register 1
EM		U		Current exit mode (user sets with MODE statement)
FL		O		Current CM field length
MFL		O		Maximum CM field length
MFLL		O		Maximum ECS field length
CMN		O		Last running CM field length divided by 100_8 (refer to RFL control statement in section 6)
ECN		O		Last running ECS field length divided by 100_8 (refer to RFL control statement in section 6)
PNL		B, R		Procedure nesting level: 0 Original control statements 1 Processing first level procedure . . 50 Processing 50th level procedure
DSC	X	U, O		Dayfile skipped control statement flag
EF	X	U, O		Previous error flag
EFG		U, R		Global error flag
TLE			EF	Time limit error
ARE			EF	Arithmetic error
PPE			EF	PPU abort
CPE			EF	CPU abort
MNE			EF	Monitor call error
ODE			EF	Operator drop
PSE			EF	Program stop error
FLE			EF	File limit error
ECE			EF	ECS parity error
TKE			EF	Track limit
MSE			EF	Equivalent to track limit
PEE			EF	CPU parity error exit
SYE			EF	System abort
FSE			EF	Forced error
ORE			EF	Override error
SSE			EF	Subsystem abort error
SRE			EF	SRU limit error
RRE			EF	Rerun error
OKE			EF	Operator kill drop
OT		O		Job origin type
SYO			OT	System origin
BCO			OT	Local batch origin
EIO			OT	Remote batch origin
TXO			OT	Time-sharing origin
SYS		O		Host operating system
NOS			SYS	NOS operating system
VER		O		Version of the operating system. This is a numeric value which varies with different systems
TIME		O		Current time of day (hhmm)

The symbolic names with true or false values are:

TRUE = 1

T = TRUE = 1

FALSE= 0

F = FALSE = 0

SWn = sense switch, n = 1 to 6

These symbolic names with true or false values, and the symbolic names in table 1-4-1 are valid in any CCL expression. They are not valid within FILE or DT functions. The FILE and DT functions have their own file function symbolic names.

CONDITIONAL STATEMENTS

The CCL conditional statements initiate conditional or unconditional skipping of statements in the control statement record of a job. There are four conditional statements: SKIP, ENDIF, IFE, and ELSE.

All conditional statements have a label string. This label string matches the statement that initiates the skip with a terminating statement that ends the skip. The terminating statement must be in the same procedure as its initiating statement.

By default, skipped control statements are not written in the dayfile of the job. The SET statement (explained later in this section) can change this default and cause skipped statements to be written in the dayfile.

SKIP STATEMENT

The SKIP statement initiates an unconditional skipping of a sequence of statements that follow the SKIP statement. Control resumes with an ENDIF terminating statement that has a label string matching the label string specified on the SKIP statement. Only an ENDIF statement can serve as a terminating statement for a SKIP statement.

The format of a SKIP statement is:

SKIP,ls.

ls Label string; 1 to 10 alphanumeric characters beginning with an alphabetic character.

An example of the use of the SKIP statement is given after the description of the ENDIF statement which follows.

ENDIF STATEMENT

The ENDIF statement terminates a skip initiated by a SKIP, IFE, or ELSE statement. In all cases, the label string on the ENDIF statement must match the label string on the statement that initiates the skip. If CCL encounters an ENDIF statement with a non-matching label string, it ignores that statement.

The format of the ENDIF statement is:

ENDIF, ls.

ls Label string; 1 to 10 alphanumeric characters beginning with an alphabetic character.

Example:

When the SKIP statement in the following sequence of control statements is processed, control skips to ENDIF, LABEL1, and none of the control statements between these two statements are processed.

```
SKIP(LABEL1)
    .
    .
any sequence of
control statements
    .
    .
ENDIF(LABEL1)
```

IFE STATEMENT

The IFE statement conditionally initiates the skipping of a group of succeeding statements. The condition is the true or false value of an expression within the IFE statement. If the expression is true, control passes to the next statement; if the expression is false, control skips to a terminating statement with a label string matching the label string specified in the IFE statement. The terminating statement must be either an ENDIF or an ELSE. If neither an ENDIF nor an ELSE statement with a matching label is found, all the remaining statements are skipped.

The format of the IFE statement is:

IFE, exp, ls.

exp A CCL expression. The separator following exp must be a comma.

ls Label string; 1 to 10 alphanumeric characters beginning with an alphabetic character.

Example:

The following control statements initiate the compilation and execution of a FORTRAN program and then test for any errors during execution. If an error was made, the error code is displayed. (The DISPLAY statement is explained later in this section.)

```
FTN,I=IFTST.
SET(EF=0)  INITIALIZE ERROR FLAG
NOEXIT.
LGO.
ONEXIT.
IFE,EF.NE.0,LABL1.
DISPLAY(EF)
ENDIF,LABL1.
```

If the program executed without error, the error flag (EF) equals zero. In this case, control passes to the ENDIF, LABL1 statement. If an error occurs, the error flag register (EF) does not equal zero, the statement is true, and control passes to the next statement; CCL then displays the error code in the error flag register.

In the following sample dayfile segment resulting from processing of the above statements, the FORTRAN program attempted to call a subroutine BETA which did not exist (outside the field length of the job).

```
16.30.35.FTN,I=IFTST.
16.30.36.       .052 CP SECONDS COMPILATION TIME
16.30.36.SET(EF=0)  INITIALIZE ERROR FLAG
16.30.37.NOEXIT.
16.30.37.LGO.
16.30.38.      NON-FATAL LOADER ERRORS -
16.30.38.  UNSATISFIED EXTERNAL REF -- BETA
16.30.38. CPU ERROR EXIT 01 AT 404254.
16.30.38.ONEXIT.
16.30.38.IFE,EF.NE.0,LABL1.
16.30.38.DISPLAY(EF)
16.30.38.      1     1B
16.30.38.ENDIF,LABL1.
```

ELSE STATEMENT

The ELSE statement acts the same as a SKIP statement unless it is placed after an IFE statement. Neither a SKIP nor an ELSE statement terminates skipping initiated by another SKIP or ELSE statement. If an ELSE statement follows an IFE statement, four possibilities of skipping can occur. These are:

- The ELSE statement has the same label as the IFE statement, and the expression in the IFE statement is true. In this case, all statements following the IFE statement up to the ELSE statement are processed. The ELSE statement then initiates a skip to the ENDIF statement with a matching label.

- The ELSE statement has the same label as the IFE statement, and the expression in the IFE statement is false. In this case, control skips to the ELSE statement and then continues processing. The ENDIF statement is ignored.

- The ELSE statement has a different label from the IFE statement, and the expression in the IFE statement is true. In this case, all statements following the IFE statement up to the ELSE statement are processed. The ELSE statement then initiates a skip to the ENDIF statement with a label matching that of the ELSE statement.

- The ELSE statement has a different label from the IFE statement, and the expression in the IFE statement is false. In this case, control skips to the ENDIF statement that has a label matching that of the IFE statement.

The format of the ELSE statement is:

ELSE(ls)

ls Label string; 1 to 10 alphanumeric characters beginning with an alphabetic character.

Example:

The following control statements test a file named TEST1 to determine if it is local to the job. (Testing with the FILE function is explained later in this section.) If the file is local, it is copied to the OUTPUT file; if it is not, it is assumed to be an indirect access permanent file, and a local copy is obtained and copied to OUTPUT.

If the test shows the file is local, each succeeding statement is processed up to the ELSE statement, which initiates a skip to the ENDIF statement. If the test shows the file is not local, control skips to the ELSE statement and each succeeding statement after the ELSE statement is processed.

```
IFE,FILE(TEST1,LO),LABEL1.
COPYSBF(TEST1,OUTPUT)
ELSE(LABEL1)
GET(TEST1)
COPYSBF(TEST1,OUTPUT)
ENDIF(LABEL1)
```

The following sample segment of a dayfile results when the above control statements are processed and TEST1 is not initially a local file.

```
11.33.00.IFE,FILE(TEST1,LO),LABEL1.
11.33.00.ELSE(LABEL1)
11.33.00.GET(TEST1)
11.33.00.COPYSBF(TEST1,OUTPUT)
11.33.01. END OF INFORMATION ENCOUNTERED.
11.33.01.ENDIF(LABEL1)
```

The following sample segment of a dayfile results when the above control statements are processed and TEST1 is initially a local file.

```
15.40.19.IFE,FILE(TEST1,LO),LABEL1.
15.40.19.COPYSBF(TEST1,OUTPUT)
15.40.21. END OF INFORMATION ENCOUNTERED.
15.40.21.ELSE(LABEL1)
15.40.21.ENDIF(LABEL1)
```

ITERATIVE STATEMENTS (WHILE AND ENDW)

The CCL iterative statements WHILE and ENDW bracket a group of control statements into a loop that can be repeatedly processed. The beginning of the loop is identified by a WHILE statement and the end by an ENDW statement. The ENDW statement must have a label string that matches the label string specified on the WHILE statement. The loop is repeated as long as the expression in the WHILE statement is true. If the expression is never true, control immediately skips to the ENDW statement; if no ENDW statement is found, all the remaining statements in the control statement record are skipped.

Label strings of all WHILE statements within the control statement record of a job should be unique. Duplication of a label string within a control statement record or within a procedure can produce unpredictable results. The same label string can be used in a called procedure and in the calling control statement record or procedure.

The formats of the WHILE and ENDW statements are:

WHILE, exp, ls.

ENDW, ls.

exp A CCL expression. The separator following exp must be a comma.

ls Label string; 1 to 10 alphanumeric characters, beginning with an alphabetic character.

Example:

The following control statements initiate a loop which is repeated five times.

```
SET(R2=5)
WHILE,R1.LT.R2,FINISH.
SET(R1=R1+1)
DISPLAY(R1)
    .
    .
ENDW,FINISH.
```

The user can vary the number of repetitions by setting different values in R2. (The SET and DISPLAY statements are explained later in this section.)

ADDITIONAL CCL STATEMENTS

SET STATEMENT

The SET statement enables the user to set the value of the following special CCL symbolic names.

Name	Description
R1	Contents of control register R1.
R2	Contents of control register R2.
R3	Contents of control register R3.
R1G	Contents of global control register 1.
EF	Error flag.
EFG	Global error flag.
DSC	Dayfile skipped control statement flag.

The SET statement format is:

SET(sym=exp)

sym	One of the symbolic names listed above.
exp	A CCL expression.

The CCL symbolic names R1, R2, R3, and R1G reference software registers which are initially zero. Only the user can set values in these registers by using an expression in the SET statement. The expression is evaluated and stored as an 18-bit, signed integer. If the expression is too large, it is truncated (retaining the sign, if signed), and no error message is issued.

The values that the user sets in these registers are available to procedures existing apart from the control statement record. (Procedures are described later in this section.)

Values set in the R1, R2, and R3 registers can be used by a called procedure, but when control returns to the caller, CCL sets R1, R2, and R3 to their values before the call. For example, PROC1 sets R1 to 1 and then calls PROC2. PROC2 sets R1 to 2 and returns to PROC1. R1 returns to its former value of 1.

A value set in the global control register, R1G, is available not only to the procedure but remains in the register when control returns to the caller.

The error flags, EF and EFG, are 6-bit, unsigned quantities initially set to zero. Either the user or CCL can change their values. Usually CCL sets the flags to error codes. It sets EF whenever an error is generated during job or procedure processing; it sets EFG during procedure exiting if EFG is zero and an error is generated during procedure processing.

If the user sets values in EF or EFG in the control statement record, these values are available to all procedures called. If the user sets values in EF or EFG in a procedure, only EFG retains the new value when control reverts from the procedure; EF returns to the value it had before the procedure was called.

If CCL encounters an error in a procedure, it sets the appropriate error code in EF but not in EFG. When control reverts, EF returns to its previous value. If EFG was zero when the error occurred, it is set to the error code when control reverts; if EFG was nonzero when the error was made, it retains its prior value.

For example, assume that EF and EFG are zero when PROC1 calls PROC2. In PROC2, an error occurs and EF is set to the appropriate error code. When control reverts to PROC1, EF is set to zero and EFG is set to the error code of the PROC2 error.

The CCL range of values for EF and EFG are system-defined numerical values for the following special symbolic names.

Symbol	Decimal Value	Description
ARE	1	Arithmetic error
CPE	4	CPU abort
ECE	15	ECS parity error
FLE	7	File limit error
FSE	10	Forced error
MNE	5	Monitor call error
MSE	8	Mass storage error (equivalent to track limit)
ODE	11	Operator drop
OKE	13	Operator kill drop
ORE	18	Override error
PEE	16	CPU parity error exit
PPE	3	PPU abort
PSE	2	Program stop error
RRE	12	Job rerun
SRE	9	SRU limit error
SSE	14	Subsystem abort
SYE	17	System abort
TKE	8	Track limit
TLE	6	Time limit error

Examples:

The following procedure file with three procedures is an indirect access permanent file with the name SETFILE.

```
.PROC,P1.
DISPLAY(R1)
DISPLAY(R1G)
SET(R1=9)
SET(R1G=888)
end-of-record
.PROC,P2.
GET(ABC)
DISPLAY(EF)
DISPLAY(EFG)
end-of-record
.PROC,P3.
GET(BASIC1)
BASIC.
DISPLAY(EF)
DISPLAY(EFG)
end-of-record
end-of-file
```

The following control statements set and display registers R1 and R1G. A procedure is called which displays these registers, resets them, and then reverts to the control statement record where they are again displayed.

```
SET(R1=1)
SET(R1G=10)
DISPLAY(R1)
DISPLAY(R1G)
BEGIN,P1,SETFILE.
DISPLAY(R1)
DISPLAY(R1G)
```

The following is a sample dayfile segment resulting from processing of the above control statements.

```
16.34.42.SET(R1=1)
16.34.42.SET(R1G=10)
16.34.43.DISPLAY(R1)
16.34.43.      1     1B
16.34.43.DISPLAY(R1G)
16.34.43.      10     12B
16.34.43.BEGIN,P1,SETFILE.
16.34.44.DISPLAY(R1)
16.34.44.      1     1B
16.34.44.DISPLAY(R1G)
16.34.44.      10     12B
16.34.44.SET(R1=9)
16.34.44.SET(R1G=888)
16.34.44.REVERT.CCL
16.34.44.DISPLAY(R1)
16.34.44.      1     1B
16.34.45.DISPLAY(R1G)
16.34.45.      888     1570B
```

The R1 and R1G registers retain their setting when the procedure is called. However, after new values are set in the procedure and control reverts to the control statement record, R1 returns to its previous value and R1G retains the value set in the procedure.

The following control statements set values in the error flags EF and EFG and then call a procedure which attempts to access an indirect access permanent file. Control reverts to the control statement record where EF and EFG are displayed to see if any error code generated is returned via these flags.

```
NOEXIT.
SET(EF=10)
SET(EFG=20)
DISPLAY(EF)
DISPLAY(EFG)
BEGIN,P2,SETFILE.
DISPLAY(EF)
DISPLAY(EFG)
```

The following sample dayfile segment results when the above statements are processed.

```
16.43.35.NOEXIT.
16.43.35.SET(EF=10)
16.43.35.SET(EFG=20)
16.43.35.DISPLAY(EF)
16.43.35.      10     12B
16.43.35.DISPLAY(EFG)
16.43.35.      20     24B
16.43.35.BEGIN,P2,SETFILE.
16.43.36.GET(ABC)
16.43.36. ABC NOT FOUND,  AT 000121.
16.43.36.DISPLAY(EF)
16.43.36.      3      3B
16.43.36.DISPLAY(EFG)
16.43.36.      20     24B
16.43.36.REVERT.CCL
16.43.37.DISPLAY(EF)
16.43.37.      10     12B
16.43.37.DISPLAY(EFG)
16.43.37.      20     24B
```

The procedure attempts to get a permanent file which does not exist. This changes EF to the error code 3. It does not affect EFG. Control reverts to the control statement record and displays EF and EFG. EF returns to its initial setting; EFG remains unchanged throughout.

To return the error code generated in a procedure to the control statement record, EFG must be zero before there is an exit from the procedure. This is demonstrated by the following control statements.

```
NOEXIT.
SET(EF=10)
BEGIN,P3,SETFILE.
DISPLAY(EF)
DISPLAY(EFG)
```

A sample dayfile segment resulting from processing of the above statements shows how the error code is returned.

```
09.42.52.NOEXIT.
09.42.52.SET(EF=10)
09.42.52.BEGIN,P3,SETFILE.
09.42.53.GET(BASIC1)
09.42.55.BASIC.
09.42.56.  INPUT FILE EMPTY OR MISPOSITIONED
09.42.56.DISPLAY(EF)
09.42.56.      4     4B
09.42.57.DISPLAY(EFG)
09.42.57.      0     0B
09.42.57.REVERT.CCL
09.42.58.DISPLAY(EF)
09.42.58.      10    12B
09.42.58.DISPLAY(EFG)
09.42.58.      4     4B
```

The procedure attempts to compile a BASIC program that is not an INPUT record. This generates an error code of 4 in EF but does not affect EFG while control is still within the procedure. When control reverts to the control statement record, EF returns to its original setting, and the error code of 4 is set in EFG.

The symbolic name DSC (dayfile skipped control statement flag) is initially zero. Under this condition, statements that are skipped during processing are not written in the dayfile. If the user sets DSC to 1, all skipped statements are written in the dayfile with two leading periods added to each statement skipped. The user can alternately set DSC to 0 and 1 any number of times. Some CCL error processing routines set DSC to 1 and force skipped statements to be written in the dayfile.

Example:

The following control statements demonstrate the effect of DSC=0 and DSC=1.

```
SET(DSC=0)
SKIP(LABL1)
COMMENT. SINCE THE DAYFILE SKIP
COMMENT. CONTROL IS SET TO ZERO,
COMMENT. THESE STATEMENTS WILL NOT
COMMENT. APPEAR IN THE DAYFILE.
ENDIF(LABL1)
SET(DSC=1)
SKIP(LABL2)
COMMENT. SINCE THE DAYFILE SKIP
COMMENT. CONTROL IS NOW SET TO ONE,
COMMENT. THESE STATEMENTS WILL
COMMENT. APPEAR IN THE DAYFILE AND
COMMENT. EACH WILL BE FLAGGED
COMMENT. WITH TWO INITIAL PERIODS.
ENDIF(LABL2)
```

The following is a sample dayfile segment resulting from processing of the preceding control statements.

```
16.49.36.SET(DSC=0)
16.49.36.SKIP(LABL1)
16.49.36.ENDIF(LABL1)
16.49.37.SET(DSC=1)
16.49.37.SKIP(LABL2)
16.49.37...COMMENT. SINCE THE DAYFILE SKIP
16.49.37...COMMENT. CONTROL IS NOW SET TO ONE,
16.49.37...COMMENT. THESE STATEMENTS WILL
16.49.37...COMMENT. APPEAR IN THE DAYFILE AND
16.49.37...COMMENT. EACH WILL BE FLAGGED
16.49.37...COMMENT. WITH TWO INITIAL PERIODS.
16.49.37.ENDIF(LABL2)
```

DISPLAY STATEMENT

The DISPLAY statement evaluates an expression and sends the result to the user dayfile in both decimal and octal integer formats. The largest decimal value which can be displayed is 10 digits. If the value is larger than 10 digits, GT followed by 9999999999 is displayed. If the value is negative and larger than 10 digits, LT followed by a minus and 9999999999 is displayed. In octal code, numbers as large as 20 digits can be displayed. For an expression larger than $2^{48}-1$, zeros are displayed.

The format of the DISPLAY statement is:

DISPLAY(exp)

exp	A CCL expression

Example:

The following sample dayfile shows several display operations.

```
15.14.59.DISPLAY(TIME)
15.14.59.      1514    2752B
15.15.07.SET(R1=99)
15.15.21.SET(R2=901)
15.15.28.DISPLAY(R1)
15.15.28.      99    143B
15.15.38.DISPLAY(R1+R2)
15.15.38.      1000    1750B
15.15.47.DISPLAY(3/2)
15.15.47.      1    1B
15.16.04.DISPLAY(2**47)
15.16.04. GT  9999999999    4000000000000000B
15.16.15.DISPLAY(-2**47)
15.16.15. LT -9999999999   -4000000000000000B
15.16.27.DISPLAY(2**48)
15.16.28.      0    0B
15.16.41.DISPLAY(99999999999)
15.16.41.  CCL156- STRING TOO LONG -  99999999999
```

The first DISPLAY statement displays the value of the TIME symbolic name. The current time given is in the form hhmm. The next six lines demonstrate the use of the R1 and R2 symbolic names. The other DISPLAY statements specify numeric expressions. The integer constant in the final DISPLAY statement has more than 10 digits, resulting in an error message.

FUNCTIONS

Functions are available for use as expressions or parts of expressions in CCL statements. These functions are not statements in themselves but must be part of a CCL statement. The CCL functions are FILE, DT, NUM, and SS.

FILE FUNCTION

The FILE function determines the attributes of a file assigned to the job.

The format of the FILE function is:

FILE(lfn, exp)

lfn	Name of the file for which attributes are being determined.
exp	An expression consisting of operators and special FILE symbolic names, which is evaluated as TRUE or FALSE for file lfn.

The parentheses and comma must be used exactly as shown in the format. The expression within a FILE function cannot include the NUM function, the SS function, or another FILE function; only the DT function or the following symbolic names can be used within the expression. Any other symbolic name within the expression is treated either as an implicit DT function (refer to the description of DT following) or as an unidentified variable.

Symbolic Name	Description
AS	File is assigned or attached to the user's control point.
BOI	File is positioned at BOI. This is valid only for a file on mass storage.
EOF	The last operation was a forward operation which encountered an EOF and is now positioned at that EOF.
EOI	The last operation was a forward operation which encountered an EOI and is now positioned at that EOI.
EQ	Equipment number where file exists.
EX	File is execute only.
ID	File ID value.
IN	File is an input file type.
LB	File is a tape which is labeled.
LI	File is a library file type.
LO	File is local file type.
MD	File has modify permission.
MS	File is on mass storage.
OP	File is opened.
PH	File is a punch file type.
PM	File is direct access permanent file type.
PR	File is a print file type.
PT	File is a primary terminal file type.
RA	File has read and allow append permission.
RD	File has read permission.
RM	File has read and allow modify permission.
TP	File is on magnetic tape.
TT	File is assigned to terminal.
WR	File has write permission.

Example:

The following sample segment from a dayfile shows the FILE function being used inside a DISPLAY statement to determine if a specified file is at BOI.

```
15.50.09.GET(ACCT)
15.50.09.DISPLAY(FILE(ACCT,BOI))
15.50.09.      1     1B
15.50.09.COPYBR(ACCT,ITEM)
15.50.09.  COPY COMPLETE.
15.50.09.DISPLAY(FILE(ACCT,BOI))
15.50.09.      0     0B
```

DT FUNCTION

The DT function determines the device type on which a file resides. DT can be used only within the expression of a FILE function. The value of the DT function is true if the 2-character mnemonic included in the function is equal to the 2-character device type. The operating system defines the mnemonics.

The format of the DT function as used in the FILE function is:

FILE(lfn,DT(dt))

lfn — Name of the file for which device residence is being determined.

dt — A 2-character mnemonic identifying the device, which may be any one of the following:

Type	Equipment
CP	415 Card Punch
CR	405 Card Reader
DE	Extended core storage
DI	844-21 Disk Storage Subsystem (half track)
DJ	844-4x Disk Storage Subsystem (half track)
DK	844-21 Disk Storage Subsystem (full track)
DL	844-4x Disk Storage Subsystem (full track)
DM	885 Disk Storage Subsystem (half track)
DP	Distributive data path to ECS
DQ	885 Disk Storage Subsystem (full track)
LP	Any line printer
LR	580-12 Line Printer
LS	580-16 Line Printer
LT	580-20 Line Printer
MS	Mass storage device

Type	Equipment
MT	Magnetic tape drive (7-track)
NE	Null equipment
NT	Magnetic tape drive (9-track)
TT	Time-sharing terminal

NUM FUNCTION

The NUM function determines if a character string is numeric or not. It evaluates the character string as true (1) if it is numeric or false (0) if it is not. NUM must be used as an expression or as part of an expression in a CCL statement.

The format of the NUM function is:

NUM(c)

c — A character string of 1 to 40 characters. If c is a literal ($-delimited), it is always nonnumeric.

SS FUNCTION

The SS function determines or sets the current subsystem being a job. SS must be used as an expression or as part of an expression in a CCL statement.

The format of the SS function is:

SS

or

SS=name

name — Identifies one of the following subsystems:

ACCESS	BATCH	FORTRAN	NULL
BASIC	EXECUTE	FTNTS	TRANACT†

The statement containing the SS function must end with a valid terminator. The SS function cannot be used in the FILE function. If it is, an error message (CCL152) is issued and the job step aborts.

The SS function is intended for use at a time-sharing terminal to determine and set subsystems by means of procedure calls.

† Not applicable to IAF.

PROCEDURES

A procedure is a group of control statements which exist apart from the control statement record of any job. (A procedure may include the time-sharing commands described in appendix E.) The purpose of a procedure is to preserve a sequence of control statements for access by a control statement record or another procedure.

A procedure is stored as a record on a file and is initiated in a job either by a CCL BEGIN statement or a call-by-name statement. This is similar to the way a program calls a subroutine. Several procedures can exist on one file and the file may be a local file, an indirect access permanent file, or an attached direct access permanent file. A procedure file can reside on magnetic tape as well as mass storage.

NOTE

A CCL procedure should not include a CLEAR statement or a NEW or OLD statement without the ND parameter. These statements return working files required by CCL when it reverts to the previous level within the job sequence of control statements.

STRUCTURE OF A PROCEDURE

A procedure consists of a procedure header statement and a procedure body. The procedure header statement must be the first line in the procedure. It names the procedure and identifies any formal keywords that can be used to transmit values to the procedure from the call statement.

The procedure body contains all statements between the header statement and the end-of-record or end-of-file. An informative error message is issued if the body does not contain at least one control statement. All control statements, including CCL statements, are legal within a procedure. The body can also include special procedure commands and data (explained later in this section).

Although a line of a procedure file can contain 150 characters, CCL interprets only the first 80 characters; it ignores characters 81 through 150. If there is no end-of-line indicator in the first 150 characters, CCL assumes that a new line begins at character 151. If the new line is not a legal CCL statement or command, it is interpreted as a control statement error when executed.

PROCEDURE HEADER STATEMENT

A procedure header statement must begin with a period followed by the characters PROC. The separators between parameters must be commas, and the header statement must be terminated by a period. Unless the header statement contains an error, it is not printed in the dayfile.

The format of the procedure header statement is:

. PROC, pname, p_1, p_2, . . . , p_n.

pname Name of the procedure; 1 to 7 alphanumeric characters. It can begin with a numeric character.

p_i Optional dummy parameters used in the body of the procedure. These dummy parameters are replaced by parameters on the procedure call statement or by default values in the procedure header. The following formats are legal.

```
fk
fk=
fk=default1
fk=default1/default2
fk=/default2
```

fk means formal keyword, an identifier that can have meaning by itself or in conjunction with other parameters. If the call statement specifies for fk, a parameter value identical to fk, then default1 is overridden and default2 is used if specified on the header (refer to Keyword Substitution in this section).

The formal keyword fk can be 1 to 10 alphanumeric characters. Any one of the defaults can be 1 to 40 characters. fk and/or the defaults can be $-delimited character strings. If a default is a $-delimited character string, it can contain special characters. fk can never contain special characters. The maximum number of keyword specifications in a procedure header statement is an installation-defined parameter. The default is 50.

Keywords can designate numeric values, variables, and file names. These appear as operands in the body of the procedure. When CCL calls a procedure, it searches the procedure body for any formal keywords declared on the header statement. [illegible]on occurs when CCL replaces formal keywords in the procedure body with default values from the header statement or values from the call statement. The parameters on the call statement determine which values replace the formal keywords.

Two special defaults can be used in a procedure header statement to identify a data record or a data file to be referenced by the procedure. These defaults are specified with a formal keyword as follows:

```
fk=≡FILE
fk=≡DATA
```

The equivalence character (≡) is the pound character (#) in ASCII. Like any other defaults in the header statement, these two can be overridden by specifications in a procedure call statement. (Use of defaults is explained in Keyword Substitution later in this section.)

The default specification fk=≡FILE in a procedure header statement associates the name specified by fk with the record that immediately follows the procedure in the same file. The user can place data in this record and reference it in the procedure with statements that include the name given to fk.

An example that illustrates procedure access of a data record is outlined in figure 1-4-1. The left side of the figure shows a file PROFIL1 with two procedures. The second procedure contains an FTN statement that gets its input from a record on a separate file INFILE. The right side of the figure shows the same operation with the FTN input coming from an additional record on the same file, PROFIL2. This additional record is the default referenced by ≡FILE. (Use of the BEGIN statement and keyword substitution is explained later in this section.)

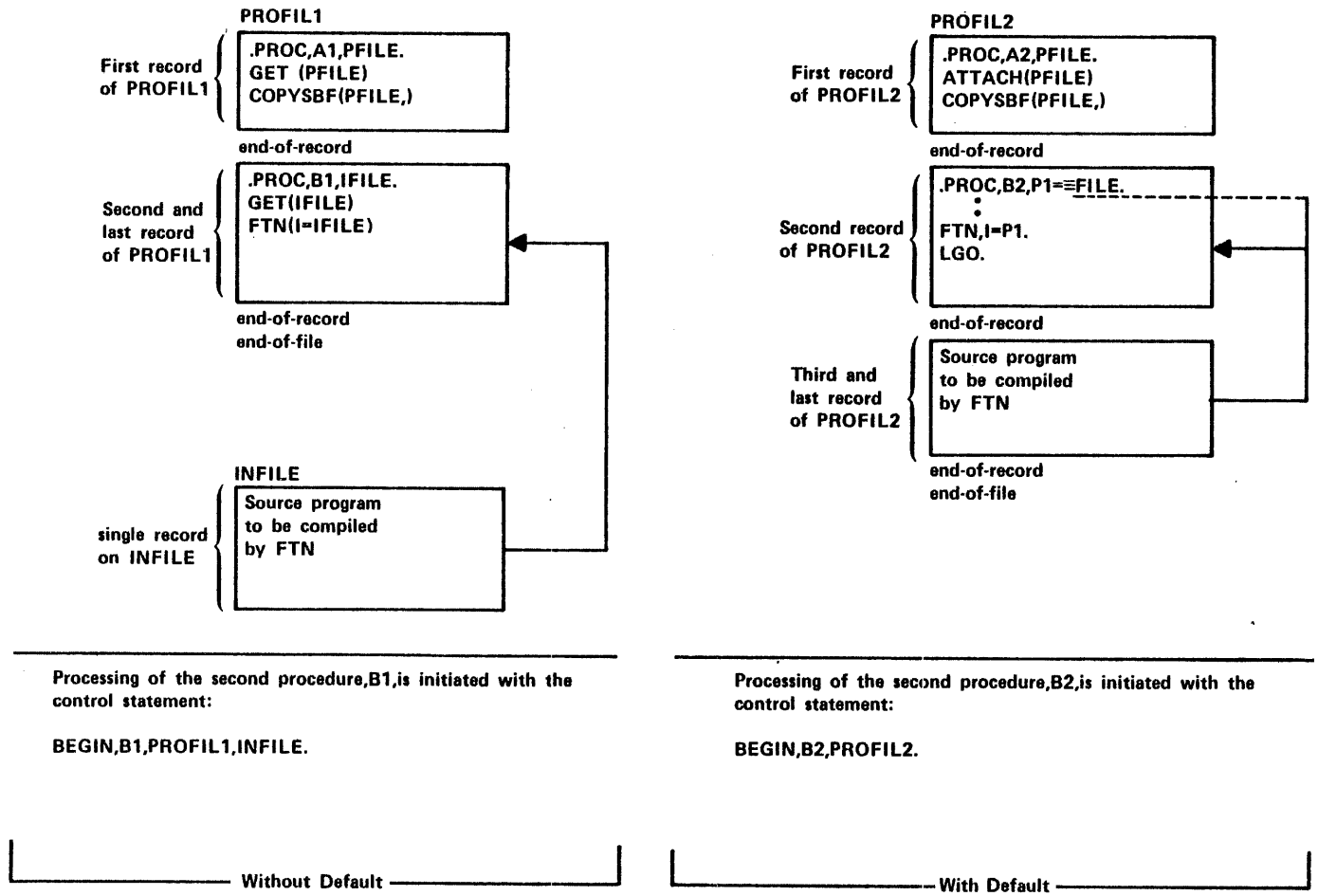


Figure 1-4-1. Procedure Access of a Data Record

The user can format a data file within the procedure itself using CCL procedure commands (refer to Procedure Commands). When the procedure is called, this formatted data is written on a separate file which the procedure can reference. When the user formats this data file, he can name it or accept the system default. In the case of the default, the procedure references the data file via an fk=≡DATA keyword specification in the header statement and the inclusion of fk in a subsequent statement.

PROCEDURE BODY

The procedure body consists of all the statements between the procedure header statement and the end-of-record. These statements can be control statements, CCL statements (including calls to other procedures), and CCL procedure commands. The parameters in these statements can be a mixture of values defined in the procedure body and keywords defined in the procedure header statement. When the procedure is called, substitutions are made for the keywords, and the procedure body becomes the control statement record until a REVERT is encountered (substitution and the REVERT statement are explained later in this section).

≡ (# in ASCII) can be placed immediately before a keyword in a procedure statement to inhibit substitution in that keyword. If two such characters (≡≡) are placed immediately before a keyword, substitution takes place and one ≡ is retained. If a single ≡ is used with a nonkeyword, it has a null effect since no substitution takes place anyway. If ≡≡ is used before such a parameter, one ≡ is retained. The ≡ does not affect a separator.

The right arrow (→) is the underline character (_) in ASCII. It can be used in a procedure statement to make a preliminary separation of two parameters (keyword or nonkeyword). After possible substitutions are made, the two parameters are joined into one. An ≡ before a → retains the → and allows substitution. A → before an ≡ does not affect the inhibiting action of the ≡.

Representative alterations of parameters in a procedure body by use of an ≡ and a → are shown in table 1-4-2.

TABLE 1-4-2. ALTERATIONS OF PARAMETERS IN A PROCEDURE BODY BY USE OF ≡ AND →

Call statement: BEGIN,APROC,APROCFL.		
Procedure header: .PROC,APROC,FK1=X, FK2=Y.		
Procedure Parameters Before Substitution	Procedure Parameters After Substitution	Comment
≡FK1,FK1	FK1,X	≡ inhibits substitution in a keyword that immediately follows.
I,J	I,J	
FK1≡FK2	XFK2	
I≡J	IJ	
≡≡FK1,FK1	≡X,X	≡≡ allow substitution if a keyword immediately follows and retains one ≡ .
≡≡I,J	≡I,J	
≡≡FK1=FK2	≡X=Y	
FK1≡,FK1	X,X	≡ does not affect a separator.
FK1→FK2	XY	The → separates two parameters before substitutions are made; after all substitutions are made, they are joined into one parameter.
I→J	IJ	
FK1→J	XJ	
I→FK2	IY	
≡→FK1,FK1	→X,X	An ≡ before a → retains the → and allows substitution.
≡→FK1=FK2	→X=Y	
FK1→≡FK1	XFK1	A → before an ≡ does not affect the inhibiting action of the ≡ .

PROCEDURE COMMANDS

Procedure commands enable the user to format a data file within a procedure and to insert documentary comments within a procedure. The commands are in fixed format with a period in column 1 and the command name beginning in column 2. A terminator must not be used, and nothing else can appear on the same line except the format specifications.

.DATA Command

A .DATA command in a procedure specifies the beginning of a sequence of data statements that are written on a separate file when the procedure is called. This data file includes all the statements, data, and file marks generated by CCL commands up to one of the following.

- Another .DATA command.
- A system end-of-record (not an .EOR command).
- A system end-of-file (not an .EOF command).
- A system end-of-information.

The data file does not include the .DATA command. Parameter substitution continues within the data statements.

The procedure can reference this separate data file with either a user-given name or the default ≡DATA. The default ≡DATA refers to a temporary file ZZCCLAx. ≡DATA is equated to a formal keyword in the procedure header. All statements that refer to the data file specify the formal keyword.

The formats of the .DATA command are:

```
.DATA

.DATA,lfn
```

The first format writes the data statements on the default file. The second writes the data statements on the file lfn. If lfn already exists, it is returned, and a new file is created. Hence, the .DATA command cannot add data to an existing file. After the data file is written, it is automatically rewound.

Example:

The following procedure file is an indirect access permanent file called DATAFIL.

```
.PROC,ALPHA,P1=≡DATA,X=FTNOUT.
FTN(I=P1,L=X)
LGO.
REPLACE(X=LISTFIL)
.DATA
      PROGRAM TEST(OUTPUT)
            •
            •
         FORTRAN source
           program
            •
            •
      END
```

This procedure file is accessed with the following call statement in a control statement record of the job.

```
BEGIN,ALPHA,DATAFIL.
```

A sample of a resulting dayfile is:

```
09.29.09.BEGIN,ALPHA,DATAFIL.
09.29.10.FTN(I=ZZCCLAA,L=FTNOUT)
09.29.11.       .047 CP SECONDS COMPILATION TIME
09.29.11.LGO.
09.29.14.    STOP
09.29.14.       .038 CP SECONDS EXECUTION TIME
09.29.14.REPLACE(FTNOUT=LISTFIL)
09.29.14.REVERT.CCL
```

All input after the .DATA command (the FORTRAN source program) is written onto the default temporary file ZZCCLAA.

.EOR Command

The . EOR command is used to separate records in a data file originating in a procedure. Whenever an . EOR is placed, an actual end-of-record is recorded when the data file is written on ≡DATA or lfn. Since the data statements are written on an external file, the . EOR command has no effect on the system end-of-record that terminates the procedure. The . EOR command is valid only after a . DATA command.

.EOF Command

The . EOF command generates an end-of-file on the data file originating in a procedure. An actual end-of-file is recorded when the data statements are written on ≡DATA or lfn. This command has no effect on the end-of-record that terminates the procedure. If the end of the data file format is also the end of the procedure, no . EOF command is needed. In this case, an end-of-record mark is added. If the user wants an end-of-file mark, he must include an . EOF command. The . EOF command is valid only after a . DATA command.

.* Command

The .* command enables the user to document a procedure with internal comments. These comments appear when the file is copied to output or displayed at a terminal; they do not appear in the dayfile when the procedure is processed. The comment, which follows the *, can contain any combination of characters.

An example of a data file written from a procedure on a named file is shown in figure 1-4-2.

PROCEDURE CALL AND EXIT

A job can call a procedure with the CCL BEGIN statement. The call inserts the procedure in the control statement record of the job following the BEGIN. The end of the procedure is signaled by the REVERT statement which CCL or the user supplies.

Figure 1-4-3 outlines the calling of a procedure from a job. Once a procedure has been called, it in turn can call a second procedure. The calling of a second procedure by a first procedure is outlined in figure 1-4-4.

BEGIN Statement

The formats of the BEGIN statement are:

BEGIN, pname, pfile, $p_1, p_2, \ldots, p_n$.

BEGIN, , pfile, $p_1, p_2, \ldots, p_n$.

pname, $p_1, p_2, \ldots, p_n$.

pname — The procedure name as declared on the header statement. This is a positional parameter.

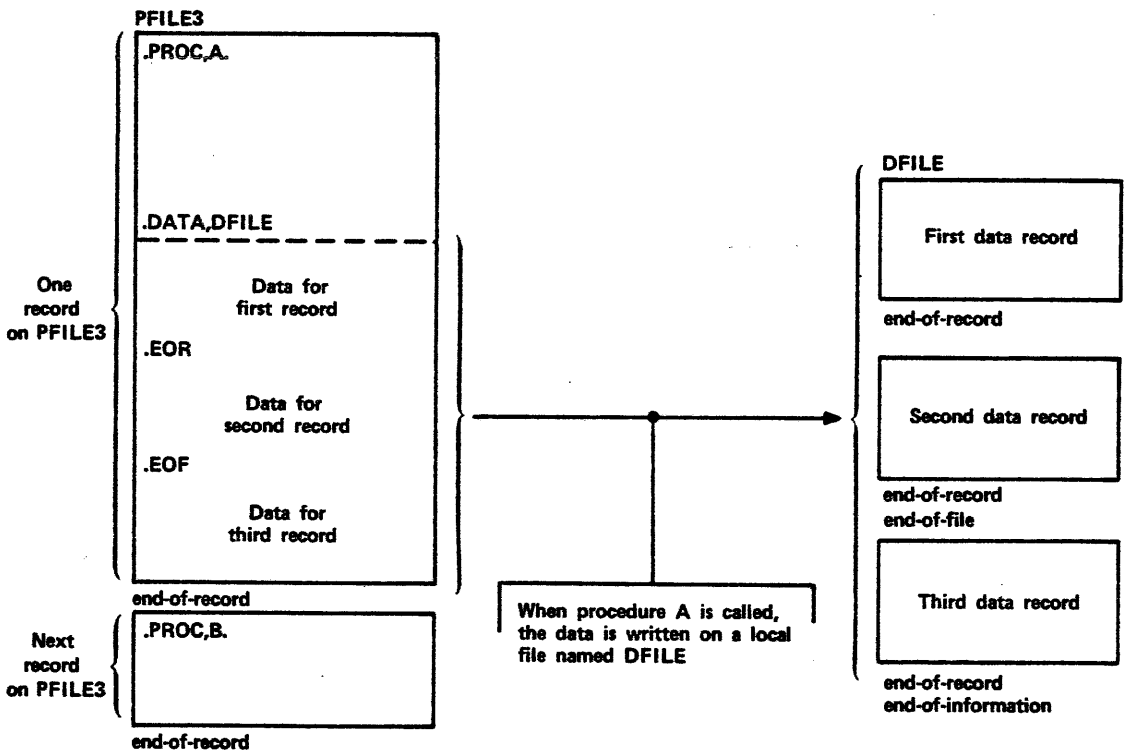


Figure 1-4-2. Data File Written from a Procedure on a Named File

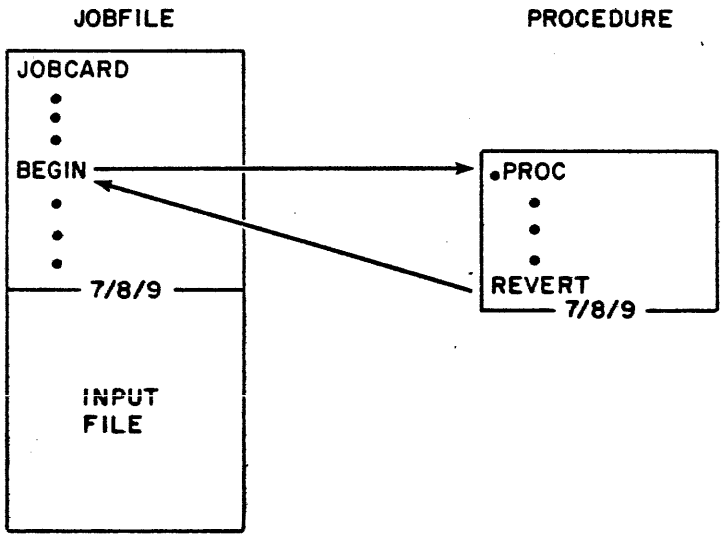


Figure 1-4-3. Calling a Procedure from a Job

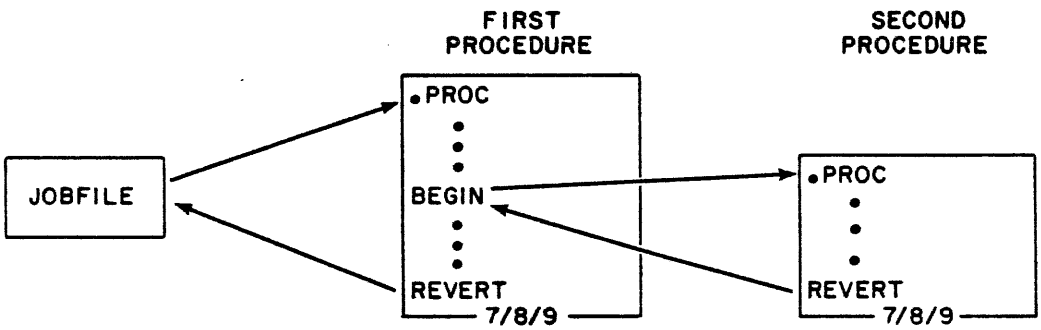


Figure 1-4-4. Calling a Procedure from Another Procedure

pfile — The name of the file on which pname is located. When the BEGIN statement is processed, CCL looks for a local file with the name pfile. Failing in this, it looks for an indirect access permanent file with that name and gets a local copy. If pfile is a direct access permanent file, the user must attach it to the job before the BEGIN statement is processed. This is a positional parameter.

p_i — Parameters that can be any of the following forms.

- fk — A formal keyword that is the same as a keyword used in the procedure header statement.
- fk= — Specifies null substitution for the formal keyword fk which appears in the procedure header statement.
- v — 1- to 40-character symbol or value that does not match any keyword in the procedure header. The symbol or value is positionally equated to a keyword in the header statement. Nonalphanumeric characters (other than /) must be within a literal ($-delimited).
- fk=v — The value v is substituted for the formal keyword, fk, which appears in the procedure header statement.

In all forms, the first separator can be a comma or a left parenthesis; the remaining separators must be commas. The terminator can be a period or a right parenthesis.

The second format uses the default procedure name which is the record at the current position of pfile. If this default is used and the file is at end-of-information, CCL rewinds pfile and calls the first procedure. An exception is the case where the procedure file is the INPUT file.

The first or second format can use the default for the file name which the installation defines (PROCFIL is the default). If pfile is specified, CCL searches for a local file with that name. Failing that, it issues a GET request to obtain a local copy of an indirect access permanent file with that name.

The third form is used to call a local file or a procedure in the system library. In this case, the procedure name must be the same as the name of the file on which it resides.

In the keyword specification fk=v, the parameter v is a user-supplied name or value that is substituted in the procedure wherever the keyword fk occurs. Special versions of this specification enable the user to substitute either the current numerical values of any of the symbolic names listed in table 1-4-1 or a boolean value. These special versions are:

fk=sym

fk=sym+

fk=sym+D

fk=sym+B

In the first form, sym is itself the numeric value which the user supplies. If the user selects any of the symbolic names in table 1-4-1 for sym in this first form, the name is substituted and not the numerical value associated with that name. For the remaining three forms with the plus symbol, any of the symbolic names listed in table 1-4-1 or a boolean value can be used. (A user-supplied name with a plus is flagged as an error.) These plus forms substitute the current numerical value associated with the symbolic name and not the name itself. The +D and +B specify decimal and octal evaluations.

The following demonstration procedure is accessed by a sequence of calling statements in the control statement record of the job.

```
.PROC,TEST1,FK.
COMMENT.    FK
```

The resulting dayfile shows each calling statement and the evaluations made. The relevant segment of the dayfile is as follows:

```
16.00.44.BEGIN,TEST1,FKTEST,FK=20.
16.00.45.COMMENT.    20
16.00.45.REVERT.CCL
16.00.46.BEGIN,TEST1,FKTEST,FK=PNL.
16.00.47.COMMENT.    PNL
16.00.47.REVERT.CCL
16.00.47.BEGIN,TEST1,FKTEST,FK=PNL+.
16.00.48.COMMENT.    1
16.00.49.REVERT.CCL
16.00.50.SET(R1=10)
16.00.50.BEGIN,TEST1,FKTEST,FK=R1.
16.00.51.COMMENT.    '
16.00.52.REVERT.CCL
16.00.52.BEGIN,TEST1,FKTEST,FK=R1+.
16.00.53.COMMENT.    10
16.00.53.REVERT.CCL
16.00.53.SET(R2=100)
16.00.54.BEGIN,TEST1,FKTEST,FK=R2+.
16.00.54.COMMENT.    100
16.00.55.REVERT.CCL
16.00.55.SET(R3=1000)
16.00.56.BEGIN,TEST1,FKTEST,FK=R3+D.
16.00.58.COMMENT.    1000
16.00.59.REVERT.CCL
16.00.59.BEGIN,TEST1,FKTEST,FK=R3+B.
16.01.00.COMMENT.    1750
16.01.00.REVERT.CCL
```

REVERT Statement

The REVERT statement terminates procedure processing. The formats are:

```
REVERT.comment
REVERT,ABORT.comment
```

comment — Character string appended after the statement terminator. This comment is especially useful to the time-sharing user because, when the REVERT statement is displayed at the terminal, following procedure processing, the comment can inform the user as to how the procedure reverted.

The REVERT statement returns control to the statement following the BEGIN statement that called the procedure. The REVERT,ABORT statement sets the error flag EF=CPE (CPU abort). Unless a NOEXIT has been processed, control goes to the next EXIT statement in the control statement record (refer to Exit Processing in section 5).

CCL always appends the following control statements to a procedure record.

```
REVERT.CCL
EXIT.CCL
REVERT,ABORT.CCL
```

These statements terminate CCL processing if no user REVERT statements are processed.

Example:

The following procedure (REVTST) is on a file called PROCFL. It reverts to the job calling it if the named file has no read permission and gives control to the job EXIT statement if the named file has no read/modify permission.

```
.PROC,REVTST,LFN1,LFN2.
IFE,FILE(LFN1,RD),LABEL1.
TDUMP(I=LFN1)
ELSE(LABEL1)
REVERT.NO READ PERMISSION
ENDIF,LABEL1.
IFE,FILE(LFN1,RM),LABEL2.
COPY(LFN2,LFN1)
ELSE(LABEL2)
REVERT,ABORT. NO READ/MODIFY PERMISSION
ENDIF,LABEL2.
```

The following two jobs (REVJOB1 and REVJOB2) call the REVTST procedure. REVJOB1 attaches an execute-only file; REVJOB2 attaches a read and/or execute file.

```
REVJOB1.
USER(USERNUM,PASWD,FAMNAME)
CHARGE(CHARGNUM,PROJNUM)
ATTACH(FILE1/UN=ALTUSER,PW=PW1,M=E)
BEGIN,REVTST,PROCFL,FILE1,XFIL.
COMMENT. RETURNS HERE
EXIT.
COMMENT. EXIT ON ERROR
```

```
REVJOB2.
USER(USERNUM,PASWD,FAMNAME)
CHARGE(CHARGNUM,PROJNUM)
ATTACH(FILE2/UN=ALTUSER,PW=PW2,M=R)
BEGIN,REVTST,PROCFL,FILE2,XFIL.
COMMENT. RETURNS HERE
EXIT.
COMMENT. EXIT ON ERROR
```

The following are the dayfiles produced by REVJOB1 and REVJOB2. REVJOB1 processes the REVERT. statement and terminates normally. REVJOB2 processes the REVERT,ABORT. statement and terminates via error processing.

```
10.09.51.REVJOB1.
10.09.51.USER(USERNUM,PASWD,FAMNAME)
10.09.51.CHARGE(CHARGNUM,PROJNUM)
10.09.51.ATTACH(FILE1/UN=ALTUSER,PW=,M=E)
10.09.52.BEGIN,REVTST,PROCFL,FILE1,XFIL.
10.09.53.IFE,FILE(FILE1,RD),LABEL1.
10.09.53.ELSE(LABEL1)
10.09.53.REVERT.NO READ PERMISSION
10.09.53.COMMENT. RETURNS HERE
10.09.54.EXIT.
10.09.54.UEAD,       0.002KUNS.
10.09.54.UEPF,       0.020KUNS.
10.09.54.UEMS,       0.229KUNS.
10.09.54.UECP,       0.049SECS.
10.09.54.AESR,       2.078UNTS.
10.21.58.UCLP, 6233,      0.128KLNS.


10.10.11.REVJOB2.
10.10.11.USER(USERNUM,PASWD,FAMNAME)
10.10.11.CHARGE(CHARGNUM,PROJNUM)
10.10.11.ATTACH(FILE2/UN=ALTUSER,PW=,M=R)
10.10.12.BEGIN,REVTST,PROCFL,FILE2,XFIL.
10.10.14.IFE,FILE(FILE2,RD),LABEL1.
10.10.14.TDUMP(I=FILE2)
10.10.14. TDUMP COMPLETE.
10.10.14.ELSE(LABEL1)
10.10.14.ENDIF,LABEL1.
10.10.15.IFE,FILE(FILE2,RM),LABEL2.
10.10.15.ELSE(LABEL2)
10.10.16.REVERT,ABORT. NO READ/MODIFY PERMISSION
10.10.16.EXIT.
10.10.16.COMMENT. EXIT ON ERROR
10.10.16.UEAD,       0.002KUNS.
10.10.16.UEPF,       0.020KUNS.
10.10.16.UEMS,       0.303KUNS.
10.10.16.UECP,       0.059SECS.
10.10.16.AESR,       2.103UNTS.
10.22.03.UCLP, 6233,      0.192KLNS.
```

KEYWORD SUBSTITUTION

Once the user has created a procedure, he specifies which keyword substitutions are made in the procedure body each time it is called by the values he includes in the list of the calling statement. When a call is made to a procedure, CCL compares the formal keywords in the call statement with the formal keywords on the procedure header statement. It then makes replacements in the procedure body either with default values from the header statement or with values specified in the call statement. The user should be aware that after substitutions are made in the procedure body, it is possible that some control statements may be expanded beyond 80 characters. For most control statements, this is flagged as an error. Exceptions are the ASSIGN, BLANK, LABEL, REQUEST, and VSN statements which allow one continuation line. The parameters on the call statement determine the values to be used as substitutions in the procedure body. CCL statements can be split between two lines if the split is at a separator.

The basic options a user has in specifying substitutions in a procedure body are listed in table 1-4-3. The user's choice is influenced by the mode in which formal keywords in the calling statement list are processed. There are two modes in which CCL processes these keywords, positional and equivalence. In positional mode, there is a one-to-one serial matching of call statement keywords with keywords on the procedure header statement to determine the substitutions to be made in the procedure body. In equivalence mode, each call statement keyword is matched with the identical alphanumeric keyword in the procedure list regardless of its position in either list.

TABLE 1-4-3. BASIC SUBSTITUTIONS IN A PROCEDURE

Formal Keyword on Procedure Header Statement	Parameter Substitutions in Procedure Body†
fk	Null, v, fk
fk=	Null, v, fk
fk=d	Null, v, fk, d
fk=d1/d2	Null, v, d1, d2

† Definitions of substitutions available:

fk	A formal keyword.
d	A single default.
d1	The first of two defaults.
d2	The second of two defaults.
null	The substitution for the keyword skipped.
v	A value other than a keyword on the header statement.

Processing of the call statement list always begins in positional mode. The switch from positional mode to equivalence mode can occur in two ways.

- An equivalence of the form fk= or fk=v appears in the call list. This initiates equivalence mode for that position and all those following in the list.
- A double default of the form fk=default1/default2 appears in the procedure header statement. The corresponding position in the call statement and all positions following are processed in equivalence mode.

Once the switch has been made from positional to equivalence mode, processing remains in that mode to the end of the list; there can be no return to positional mode in that call.

The permissible call statement parameters in positional mode are:

- A keyword identical to a keyword in the procedure header statement. This keyword can be $-delimited.
- A value that does not match any keyword in the procedure header statement. This value can be $-delimited.
- An omitted entry indicated by double commas or by the call list being shorter than the procedure list.

In the first case, the call statement keyword overrides all procedure defaults except a double default. For the double default, the second is used.

In the second case, the call statement value overrides all procedure defaults except a double default. The double default is an error since it initiates equivalence mode in the corresponding position of the call statement, and in that mode, the call cannot use a value that is unidentified in the procedure header statement.

In the third case, the procedure defaults are used. This includes the first default of a double default.

These substitutions are summarized in table 1-4-4.

The permissible call statement parameters in equivalence mode are:

- A keyword identical to a keyword in the procedure header statement. This keyword can be $-delimited.
- A keyword identical to a keyword in the procedure header statement that is equivalenced to null. This keyword can be $-delimited.
- A keyword identical to keyword in the procedure header statement that is equivalenced to a value not in the header statement. This keyword can be $-delimited.
- An omitted entry indicated by double commas or by the call list being shorter than the procedure header list.

TABLE 1-4-4. KEYWORD SUBSTITUTION IN POSITIONAL MODE

Procedure Header Statement	Keyword on the BEGIN Statement				
	Omit	fk	fk	v	v
.PROC,,fk.	fk	fk	fk	v	v
.PROC,,fk=.	Null	fk	fk	v	v
.PROC,,fk=d.	d	fk	fk	v	v
.PROC,,fk=d1/d2.	d1	d2	d2	Error	Error
.PROC,,fk.	fk	fk	fk	v	v
.PROC,,fk=.	Null	fk	fk	v	v
.PROC,,fk=d.	d	fk	fk	v	v
.PROC,,fk=d1/d2.	d1	d2	d2	Error	Error

In the first case, the procedure header defaults are substituted. The second default of the double default is used.

In the second case, the null string is substituted.

In the third case, the value overrides all defaults in the procedure header statement.

In the fourth case, the procedure defaults are substituted. The first default of the double default is used.

These substitutions are summarized in table 1-4-5.

TABLE 1-4-5. KEYWORD SUBSTITUTION IN EQUIVALENCE MODE

Procedure Header Statement	Keyword on the BEGIN Statement				
	omit	fk or fk	fk= or fk=	fk=v or fk=v	v or v †
.PROC,,fk.	fk	fk	Null	v	Error
.PROC,,fk=.	Null	Null	Null	v	Error
.PROC,,fk=d.	d	d	Null	v	Error
.PROC,,fk=d1/d2.	d1	d2	Null	v	Error
.PROC,,fk.	fk	fk	Null	v	Error
.PROC,,fk=.	Null	Null	Null	v	Error
.PROC,,fk=d.	d	d	Null	v	Error
.PROC,,fk=d1/d2.	d1	d2	Null	v	Error

Examples:

The following four examples match a calling statement with the header statement of the procedure called. The parameters are spaced to illustrate positional correlation.

```
                        positional | equivalence
BEGIN,AA,PFILE, A, B, C, | Z=D.
.PROC,AA,        W, X, Y, | Z.

                   positional | equivalence
BEGIN,BB,PFILE, A, | X,      Y=C, Z=D.
.PROC,BB,       W, | X=A/B, Y,   Z  .

                   | equivalence
BEGIN,CC,PFILE, | W=A, X=B, Y=C, Z=D.
.PROC,CC,       | W,   X,   Y,   Z  .

                   | equivalence
BEGIN,DD,PFILE, | Z=D, Y=C, X=B, W=A.
.PROC,DD,       | W,   X,   Y,   Z  .
```

If the following statement were in each of the above procedures,

```
COMMENT.      W   X   Y   Z
```

it would, in all four cases, read as follows after substitution.

```
COMMENT.      A   B   C   D
```

† Assumes the parameter is entered under equivalence mode.

The following sequence of BEGIN statements is included in the control statement record of the job. These reference two files, SUBP and SUBE, that have procedures.

```
BEGIN,,SUBP.
BEGIN,,SUBP,P1,P2,P3,P4.
BEGIN,,SUBP,B1,B2,B3.
BEGIN,,SUBP,P1=B1,P2=B2,P3=B3,P4=B4.
BEGIN,,SUBE.
BEGIN,,SUBE,P1,P2,P3,P4.
BEGIN,,SUBE,P1=,P2=,P3=,P4=.
BEGIN,,SUBE,P1=S1,P2=S2,P3=S3,P4=S4.
```

The file SUBP is as follows:

```
.PROC,,P1,P2=,P3=DEF,P4=DEF1/DEF2.
COMMENT.    P1    P2    P3    P4
```

The file SUBE is as follows:

```
.PROC,,P4=DEF1/DEF2,P1,P2=,P3=DEF.
COMMENT.    P1    P2    P3    P4
```

The following is a segment of the dayfile that results when the BEGIN statements are processed.

```
12.58.01.BEGIN,,SUBP.
12.58.02.COMMENT.    P1         DEF    DEF1
12.58.02.REVERT.CCL
12.58.03.BEGIN,,SUBP,P1,P2,P3,P4.
12.58.04.COMMENT.    P1    P2    P3    DEF2
12.58.04.REVERT.CCL
12.58.05.BEGIN,,SUBP,B1,B2,B3.
12.58.06.COMMENT.    B1    B2    B3    DEF1
12.58.06.REVERT.CCL
12.58.07.BEGIN,,SUBP,P1=B1,P2=B2,P3=B3,P4=B4.
12.58.09.COMMENT.    B1    B2    B3    B4
12.58.09.REVERT.CCL
12.58.10.BEGIN,,SUBE.
12.58.13.COMMENT.    P1         DEF    DEF1
12.58.13.REVERT.CCL
12.58.14.BEGIN,,SUBE,P1,P2,P3,P4.
12.58.14.COMMENT.    P1         DEF    DEF2
12.58.14.REVERT.CCL
12.58.15.BEGIN,,SUBE,P1=,P2=,P3=,P4=.
12.58.16.COMMENT.
12.58.16.REVERT.CCL
12.58.16.BEGIN,,SUBE,P1=S1,P2=S2,P3=S3,P4=S4.
12.58.17.COMMENT.    S1    S2    S3    S4
12.58.19.REVERT.CCL
```

The substitutions made in a procedure that calls a second procedure is shown in figure 1-4-6. The resultant dayfile is shown on the right side of the figure.

```
GET(PROGRM1)
BEGIN,EXECUTE,PFILE1,PROGRM1,PRINT.
```

PFILE1

```
.PROC,EXECUTE,NAME,OUT.
FTN(I=NAME,L=OUT)
LGO.
IFE,EF=0,DROP.
BEGIN,LISTING,PFILE2,OUT.
ENDIF,DROP.
```

PFILE2

```
.PROC,LISTING,OUTFILE=OUT.
REWIND(OUTFILE)
COPYSBF(OUTFILE,OUTPUT)
```

RESULTANT DAYFILE

```
16.01.08.GET(PROGRM1)
16.01.08.BEGIN,EXECUTE,PFILE1,PROGRM1,PRINT.
16.01.09.FTN(I=PROGRM1,L=PRINT)
16.01.10.       .043 CP SECONDS COMPILATION TIME
16.01.10.LGO.
16.01.11.     STOP
16.01.11.       .038 CP SECONDS EXECUTION TIME
16.01.12.IFE,EF=0,DROP.
16.01.12.BEGIN,LISTING,PFILE2,PRINT.
16.01.12.REWIND(PRINT)
16.01.13.COPYSBF(PRINT,OUTPUT)
16.01.13. END OF INFORMATION ENCOUNTERED.
16.01.13.REVERT.CCL
16.01.13.ENDIF,DROP.
16.01.13.REVERT.CCL
```

Figure 1-4-5. Keyword Substitution in Two Procedures

CONTROL STATEMENT PROCESSING 5

Jobs entering the system consist of one or more logical records. The first logical record contains system directives (control statements) which describe the processing that is to occur in the job file (job deck). This section describes control statement processing and how the control statements affect other aspects of job processing.

The operating system recognizes three types of control statements.

- Local File Control Statements — These statements call files that are assigned to the job control point. LGO is the system default local file used for retaining object code generated by one of the language processors.
- System Control Statements — These statements are divided into eight categories.
  - Job control control statements
  - File management control statements
  - Permanent file control statements
  - Load and dump central memory utility control statements
  - Tape management control statements
  - System utility control statements
  - Library utility control statements
  - Loader control statements†
- Product Set Control Statements — The product set control statements call the various products available under NOS. Their formats are given in the applicable product reference manual and in the Applications Programmer's Instant.

CONTROL STATEMENT FORMAT

All control statements may consist of one to four fields. The first field is the statement label field. If present (the field is optional), it begins with a numeric character and terminates with a separator character. The field is used only in conjunction with the system control language described in appendix H.

† Refer to the CDC CYBER Loader Reference Manual.

The second field, also optional, is a $ or / prefix character which precedes the program name. If a $ is present, it indicates that the specified program to be executed must be loaded from the system library.† Therefore, even if a local file of the same name is present, the system program, not the local program, is executed.

The / option may be used on local file control statement calls. If a / is present, it indicates that the parameters following the program name are to be processed in the operating system format. If a / is not present, the parameters are processed in product set format. The default is product set format because most programs specified in local file calls have been generated by one of the product set members. The / option does not apply for control statement calls to programs residing on the system library. For those types of calls, parameters are processed in the operating system format unless the SC directive to SYSEDIT has been entered. Refer to the SYSEDIT control statement in the System Maintenance Reference Manual for a description of the SC directive.

The third field contains the name of the program to be executed. The fourth field (optional) contains parameters which further define the operation to be performed. The parameter field is set off from the name field by a separator character. A valid terminator character must follow the fourth field (or the third field if no parameters are present).

The system allows continuation lines for ASSIGN, BLANK, LABEL, REQUEST, and VSN control statements. (For details, refer to Control Statement Rules in section 10.)

The following is a comparison of the operating system and product set formats (refer to the Applications Programmer's Instant for control statements using the product set format).

Operating System Format	Product Set Format
1. Valid separators are + - " / = , (and any other character with a display code value greater than 44_8 except *) $. and blank.	1. Same as for the operating system format.
2. Valid terminators are .)	2. Same as for the operating system format.
3. Letters, numbers, and the * are the only characters allowed in the parameter field. The one exception to this rule is the use of literals (that is, character strings delimited by dollar signs). Characters other than letters, numbers, and the * can be included in literals. No characters within a literal have special meanings; the system merely checks the syntax of the literal. The called program must do its own processing of the literal. Literals are allowed only on equipment/file assignment control statements and loader control statements.	3. Any parameter field that includes characters other than letters, numbers, and the * must be expressed as a literal.

Operating System Format	Product Set Format
4. All embedded blanks within a control statement except those appearing in literals are ignored.	4. All embedded blanks within a control statement except those appearing in literals or after the program name are ignored.
5. Comments may appear on the control statement but they must follow the terminator. They may contain any character. Comments are not printed for some control statements.	5. Same as for the operating system format.
6. Parameters, separators, and terminators are stored in the user's field length beginning at RA+2. The characters , . and) are stored as zero. For all parameters and all valid separators except the comma, their display code equivalent is stored.	6. Parameters are stored in their display code equivalent beginning at RA+2. Separators and terminators are stored as follows:

Character	Code (Octal)
,	1
=	2
/	3
(	4
+	5
-	6
;	10
) or .	17
Other valid separators	16

7. File names are 1 to 7 alphanumeric characters.	7. File names are 1 to 7 alphanumeric characters. File names beginning with a numeric character are illegal.
8. Not NOS/BE compatible	8. NOS/BE compatible

In general, no parameter can contain more than 7 characters. If a parameter contains more than 7 characters, the entire control statement is issued to the dayfile, followed by the message:

FORMAT ERROR ON CONTROL CARD.

There are two exceptions to this rule. If a statement calls a program from the system library that has an ARG= entry point, parameters in the statement can contain more than 7 characters. If a parameter contains more than 7 characters, the ARG= entry point is not present, and the SDM= entry point is present (refer to appendix F in volume 2), the statement name (such as DEFINE) is issued to the dayfile but all parameters are suppressed.

The parameters can appear in either order-dependent or order-independent format. Order-dependent parameters are required when the parameters must be passed in a specific order. An example of order-dependent parameters is:

RESEQ(MYFILE,B,,20)

In this example, the system expects the resequencing increment to be passed as the fourth parameter; therefore, a separator must be present for the parameter not specified.

Order-independent parameters may be passed in any order. This is made possible by the use of keywords. Keywords are identifiers which have meaning either by themselves or when used in conjunction with other parameters. Usually, keywords are passed with a parameter and a separator. The separator must not be a comma. When the list of parameters is passed to the called program, all separators except commas are also passed.

Some programs require specific separators (usually =), and others merely require that a separator be present. Examples of keyword notation are:

1. COBOL(I=SFILE, B=BFILE)
2. COBOL(B=BFILE, I=SFILE)
3. COBOL(L=0, A, F)
4. JOBX, T10, CM45000.

In examples 1 and 2, both parameters and separators are passed to the COBOL compiler. Since these parameters are order-independent, both statements produce the same result.

In example 3, two keywords are passed with no separator character or parameter. In example 4, the keyword is the first character of the parameter.

The parameters and an image of the control statement being processed are written in the job communication area. The job communication area is the first 110_8 words of the user's field length, from RA through RA+107_8. Appendix E in volume 2 describes the first 100_8 words of this area.

The following control statements produce the same image in CM. Both statements are processed using operating system format.

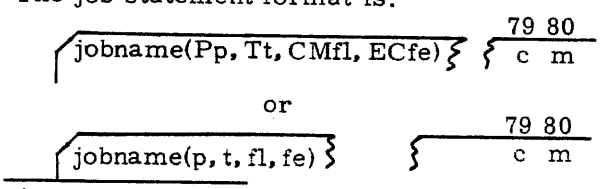
```
123, PERMIT (FILEABC, USERAAA=R, USERBBB=W)
123, $PERMIT (FILEABC, USERAAA=R, USERBBB=W)
```

JOB STATEMENT (JOB CARD)

The job statement (also known as the job card) names the job and may specify job processing parameters. The first statement of a job input file must be a job statement. †

The user can issue the job statement in order-independent or order-dependent format. In order-independent format, a separator character does not appear between the keyword and its value. If the order-dependent format is used and parameter values are omitted between separators, the parameter values are interpreted as zeros. A parameter value containing an 8 or 9 must not have a B suffix. If there is a syntax error in the job statement, the system issues an error message and terminates the job.

The job statement format is:



† Not applicable to time-sharing jobs.

jobname — Alphanumeric job name (1 to 7 characters) which must begin with a letter. This name identifies the individual jobs being run under the same user number.

Pp or p — Priority level (octal) at which the job enters the system, ranging from 1 to 17_8. A value containing an 8 or 9 or the suffix D is interpreted as decimal. This parameter is not used by NOS (refer to Job Scheduling in section 3).

Tt or — Central processor job step time limit in seconds. Values can range from 1 to 77777. Decimal is the default base. Octal values from 1 to 77777_8 (1 to 32767) can be entered if followed by a B suffix. The default limit is 64 (100_8). Decimal values from 32767 to 77777 set the time limit at its maximum. The time limit set by this parameter must be sufficient for completion of each of the steps in the job (refer to Time Limit Control in section 3).

CMfl or fl — Maximum octal field length for the job (refer to Field Length Control in section 3). The system rounds the value to the next highest multiple of 100_8. The field length cannot exceed:

377700_8 on a 198K or a 262K machine

360000_8 on a 131K machine

163000_8 on a 65K machine

A value containing an 8 or 9 or the suffix D is interpreted as decimal.

ECfe or fe — Maximum octal number of 1000_8 word ECS blocks required by the job. This ECS field length cannot exceed 3777 blocks or the amount of user ECS allowed by the installation. The user job must request the ECS (refer to the RFL control statement in section 6) before it can be used. A value containing an 8 or 9 or the suffix D is interpreted as decimal.

cm — Conversion mode entered in columns 79 and 80.† A 26 indicates that coded cards are to be converted in O26 mode; 29 indicates cards are converted in O29 mode. This initial keypunch mode can be changed within the job by a conversion change card (refer to Coded Cards in appendix F) when reading cards or a ROUTE statement when punching cards. If this parameter is omitted, the installation default keypunch mode is used.

The system issues error messages when parameter specifications exceed validation limits. It also issues an error message if an ECS field length is specified when the user's CM validation limit is less than 10000_8 words. The user should consult installation personnel for further installation restrictions based on the machine configuration and subsystems used.

† This conversion mode indicator is ineffective for remote batch jobs entered under Export/Import or mode 4 RBF; it is effective for remote batch jobs entered under HASP RBF.

Example:

```
JOBAAA,,,50000,20.
```

has the same effect as:

```
JOBAAA,0,0,50000,20.
        or
JOBAAA,P0,T0,CM50000,EC20.
```

CONTROL STATEMENT PROCESSING FLOW

The system translates a control statement by:

1. Reading the statement from the control point control statement buffer. If necessary, the system reads control statements from the job input file.

2. Deleting all spaces between the beginning of the statement and the terminator character (a period or a right parenthesis). In general, the system allows only standard FORTRAN characters to appear before the terminator character, although other characters can appear within a literal or in the comment field.

3. Comparing special control statement names with the name of the control statement being processed. If the statement name is CTIME, HTIME, RTIME, or STIME, the system processes the control statement.

4. Searching the file name table for a file assigned to the job with a name identical to the name of the control statement. However, if a $ precedes the program name, this step is skipped. If an identical name is found, the program is loaded into memory. The arguments are extracted from the control statement and stored in RA+2 through RA+n+1 (n is the number of parameters). The CPU is requested to begin execution unless special loader control statements follow.

5. Searching the central library directory for a program name that matches the control statement name. If the name is found, the system proceeds as in step 4; otherwise, the system searches further.

6. Searching the peripheral processor library directory for a program name that matches the control statement name. If found, the name is placed, with a maximum of two arguments, as a peripheral processor request, and the system exits to the program.

7. If the control statement name is not found during any of the above searches, the control statement is declared illegal and the job is aborted.

Figure 1-5-1 illustrates the flow of control statement processing.

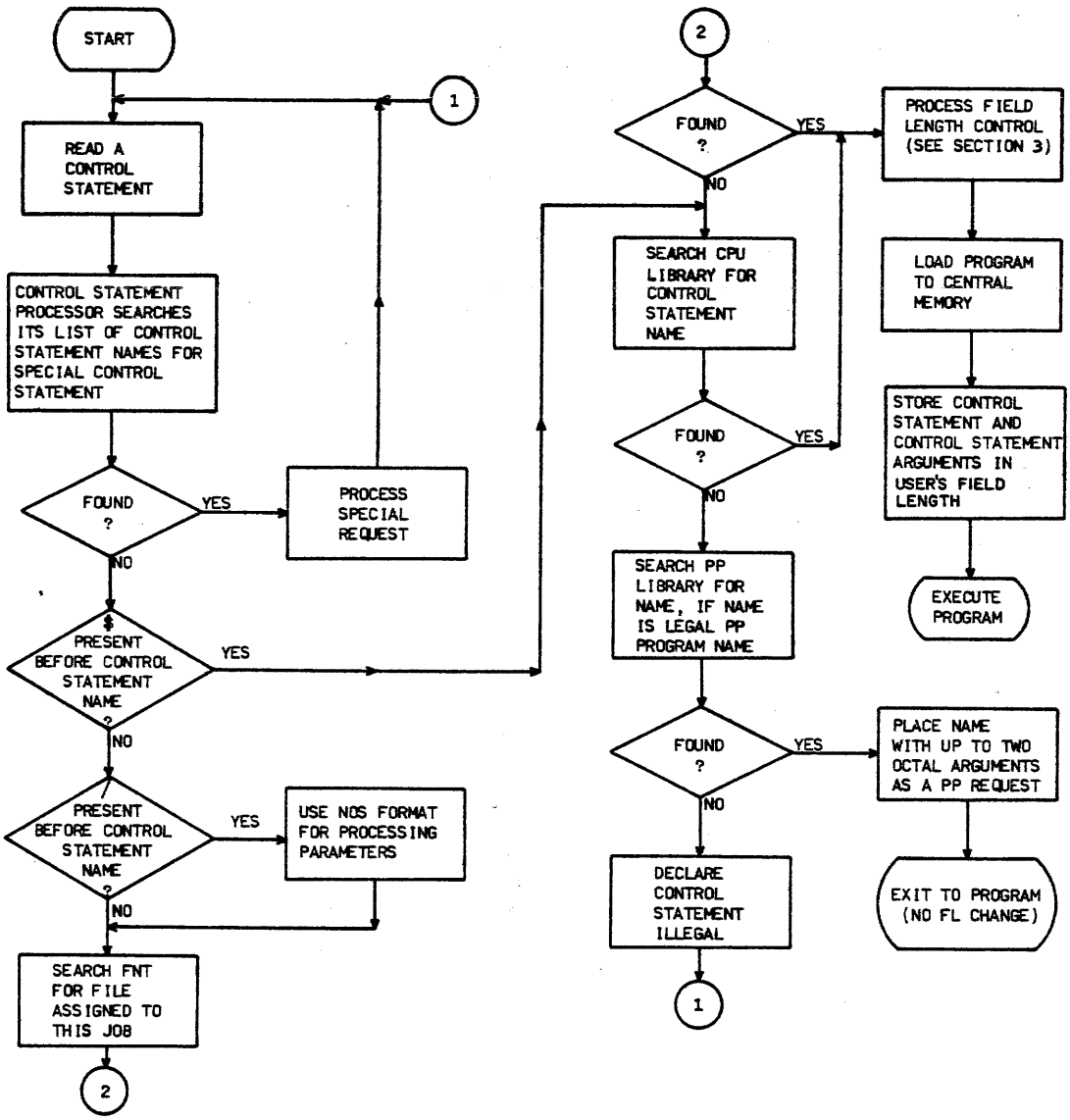


Figure 1-5-1. Control Statement Processing Flow

EXIT PROCESSING

When an error condition occurs during job processing, the system searches the control statement record for an EXIT statement. If the record does not contain an EXIT statement, the system terminates the job. If the system finds an EXIT statement, it clears the error condition and processes the control statements that follow the EXIT statement. If the error was a time limit error, the limit is reset to the time used plus 10_8 seconds. This gives the user time for post-error cleanup operations. If the error was an SRU limit error, the limit is reset to the SRUs used plus 10_8 SRUs.

If a NOEXIT statement is encountered, normal error processing is not performed. That is, if the no exit flag has been set (by the NOEXIT statement) prior to the error, the error flag is cleared, no search is made for an EXIT statement, and processing continues with the next control statement. An ONEXIT statement can be used to return to error processing mode; it clears the no exit flag.

The following sequence of control statements illustrates this exit processing.

```
JOBCCC.
USER(SMITH22,SM)
CHARGE(55A19,69P5)
NOEXIT.
GET(A,B)
ONEXIT.
ATTACH(MASTER/M=W)
SKIPEI(MASTER)
COPYBF(A,MASTER)
COPYBF(B,MASTER)
PACK(MASTER)
COPYSBF(MASTER,)
EXIT.
ENQUIRE(F)
-EOR-
-EOI-
```

This job gets local copies of two indirect access permanent files and adds them to a direct access file. The NOEXIT suspends error processing, and the job continues even if file A and/or B is not found. The ONEXIT turns error processing back on. If any error occurs thereafter, processing skips to the EXIT statement and continues with the ENQUIRE. If no error occurs after the NOEXIT, processing continues until the EXIT statement and terminates (ENQUIRE is not processed).

JOB CONTROL CONTROL STATEMENTS 6

The job control control statements enable the user to alter information that controls his job while in the system and to retrieve information concerning the status of his job. The control statements included in this category are:

ACCOUNT	MODE	ROLLOUT
CHARGE	NOEXIT	RTIME
COMMENT	NORERUN	SETASL
CTIME	NOTE	SETJSL
DAYFILE	OFFSW	SETPR
ENQUIRE	ONEXIT	SETTL
ENTER	ONSW	STIME
EXIT	PASSWOR	SUBMIT
HTIME	PROTECT	SUI
LDI	RERUN	SUMMARY
LENGTH	RESOURC	SWITCH
LIMITS	RFL	USECPU
MFL		USER

The user must have specific validation parameters set to use LDI, PASSWOR, PROTECT, SUBMIT, or SUI. He can use the remaining statements regardless of his validation. A listing of validation information can be obtained using the LIMITS statement. Although the user is allowed to change several control values for his job (such as RFL, SETPR, and SETTL), he can never specify more than that for which he is validated.

The system uses the USER statement and CHARGE statement for checking user validation and system accounting information. The RESOURC statement is used by the system to prevent deadlocks from occurring when several tapes or packs are used concurrently.

The user can submit files as batch origin type jobs through the LDI and SUBMIT control statements. He can specify the mode of error exit processing desired through use of the EXIT, ONEXIT, NOEXIT, and MODE statements. He can also set conditions for his program with sense switches (such as ONSW, OFFSW, and SWITCH). In the event of a system malfunction causing jobs to be recovered, he may either allow his job to be run again with the RERUN statement or prevent it from being rerun with the NORERUN statement. Additional information is returned to the user by the CTIME, RTIME, STIME, HTIME, and DAYFILE statements. The COMMENT statement allows the user to provide his own dayfile documentation.

ACCOUNT STATEMENT

The ACCOUNT control statement is included for compatibility with previous systems. The USER control statement should be used with the present system.

CHARGE STATEMENT

The CHARGE statement causes the system to record on the account dayfile all information regarding resources used under a specified charge number/project number combination. Its purpose is to control the accounting activity of the system for a customer or the installation.

The control statement format is:

CHARGE(chargenum, projectnum)

chargenum	A 1- to 10-alphanumeric character charge number assigned to the user.
projectnum	A 1- to 20- alphanumeric character project number assigned to the user.

For added security, the user may issue the CHARGE statement without parameters. In this case, the system reads the parameters from a record in the INPUT file. This record must be a single line with the format:

chargenum, projectnum

The CHARGE statement is used in conjunction with user accounting control. An installation which implements this feature can impose limits on the SRUs a user may accumulate or restrict his access to the system to a certain time-of-day interval.

If access option 8 is not set (refer to LIMITS control statement in this section), the user must include a CHARGE statement immediately following every USER statement in his job. If option 8 is set, the user may but is not required to include a CHARGE statement. A user assigned more than one charge and/or project number may include additional CHARGE statements in his job to record resources used under each charge number/project number combination. Whenever a new CHARGE statement is issued, the SRU information for the previous charge number/project number is written to the account dayfile and then cleared. However, the other accumulators (central processor time, mass storage activity, and so on) are not cleared but continue to increment. The following message is also issued when a new CHARGE statement is entered.

yy.mm.dd. hh.mm.ss. jobname. ACCN, chargenum, projectnum.

For a complete list of accounting messages issued to the user's dayfile, refer to Job Completion in section 3.

COMMENT STATEMENT

The COMMENT statement enters the specified comment in the system and user's dayfile.

The control statement format is:

COMMENT.comments

or

*comments

comments — Any combination of characters the user wishes to display

If the

*comments

format is used, the * must be the first nonblank character.

CTIME STATEMENT

The CTIME control statement requests that the accumulated CPU time for the job be issued to the user's dayfile (in seconds).

The control statement format is:

CTIME.

DAYFILE STATEMENT

The DAYFILE control statement causes the system to write the user's control point dayfile to the file specified.

The control statement format is:

DAYFILE(lfn, strng, op, pd, pl)

or

DAYFILE(L=lfn, FR=strng, OP=op, PD=pd, PL=pl)

L=lfn — File on which the dayfile is to be written. If omitted, OUTPUT is assumed. Pagination will occur if listing file name is OUTPUT or if PD or PL is specified.

FR=strng — This parameter specifies the literal string for which a search is to be made in the dayfile. Unless the literal string is a valid command or control statement (7 characters or less), it must be enclosed by $ delimiters. The first character of the literal string requested must always be the starting position of the field (for example, the first character of the time field is a space). The field to be searched is specified by the op parameter. The portion of the dayfile from the last occurrence of the requested literal string to the end of the dayfile is returned to the user.

OP=op — Selects search option (single character):

op	
T	Search time field for matching string.
M	Search message field for matching string.
I	Incremental dump (from point of last dump).
F	Full dump.

If a literal string (strng) is specified and op is omitted, OP=M is assumed; if both strng and op are omitted, OP=F is assumed.

PD=pd — Print density (3, 4, 6, or 8 lines per inch); if omitted, PD=6 is assumed.

PL=pl — Selects page size; if omitted, page size is determined from print density. Page size does not include title lines.

PD	Assumed PL
3	30
4	40
6	60
8	80

Examples:

```
DAYFILE(TEMP,$ABCDEFG$)
DAYFILE(L=TEMP,FR=$ABCDEFG$,OP=M)
DAYFILE(FR=COMPASS)
```

ENQUIRE STATEMENT

The ENQUIRE control statement gives information about the system to the user. Three forms of the command are allowed.

The control statement formats are:

ENQUIRE(OP=$p_1p_2...p_n$,JN=jobname,FN=lfn$_1$,O=lfn$_2$)

or

ENQUIRE($p_1p_2...p_n$)

or

ENQUIRE.

p_i — Any of the following options.

Option	Description
A	Gives listings of the B, D, R, U, J, L, and F options, respectively.

Option	Description
B	Returns identification and priority information to the user. Example: USER NUMBER DLH2500 USER INDEX HASH AKWA JOB NAME AKQAAEF JOB SEQ. NO. AAEF FAMILY CLS127 PACKNAME *NONE*. PRIMARY FILE *NONE*. SUB SYSTEM NULL. QUEUE PRIORITY 4010 CPU PRIORITY 30 MAX FL (CM) 203700 MAX FL (EC) 0 LAST FL (CM) 0 LAST FL (EC) 0
D	Returns a listing of the resources the user has demanded and those which have been assigned. Example: RESOURCE DEMAND INFORMATION. RESOURCE DEMAND ASSIGNED MT 2 2
F	Gives the status of files at the user's control point. An asterisk (*) after the file type indicates that the file is locked. (The user cannot write on a locked file.) Refer to the FILE function in section 4 for the meaning of the file type mnemonics. The STATUS column lists the last operation performed on the file. (I/C means incomplete.) Example: FILENAME LENGTH/PRUS TYPE STATUS EXAMP 2 LO. EOR READ INPUT 3 IN.* EOR READ BFILE3 21 LO. EOR READ OUTPUT 3 PR. I/C WRITE TOTAL = 4
J	Returns the contents of the user's control registers, error flag field, and succeeding control statements. Example: JOB CONTROL REGISTERS. R1 = 32 R2 = 98 R3 = 0 EF = 0 EFG = 0 R1G = 0

Option	Description

CONTROL STATEMENT(S).

```
GET(ALPHA)
COPYSBF(ALPHA,)
*EOR*
```

If the J option is used within a CCL procedure, only the remaining control statements in the procedure are listed.

L Returns user's loader information.

Example:

```
LOADER INFORMATION.
    MAP OPTIONS = SBX
    GLOBAL LIBRARY SET IS -
EMPTY.
```

R Returns to the user the amount of resources used. The resources listed include CPU time, mass storage activity, magnetic tape activity, and permanent file activity. These statistics are factors used in calculating SRUs used.

Example:

```
RESOURCES USED.

CPU TIME          0.025 SECS.
MS ACTIVITY       0.117 KUNS.
MT ACTIVITY       0.000 KUNS.
PF ACTIVITY       0.010 KUNS.
ADDER             0.002 KUNS.
SRU               2.025 UNTS.
```

S Returns the user's accumulated SRUs. The SRU represents the total usage of the system by the user. This unit is derived from central processor time, I/O activity, and memory usage.

Example:

```
SRU ACCUMULATOR.

    SRU           2.030 UNTS.
```

T Returns accumulated CPU time.

Example:

```
CPU ACCUMULATOR.

    CPU TIME      0.017 SECS.
```

U Returns the initial amount of resources available to the user in seconds, job step SRU, account block SRU, and the remaining resources available for dayfile messages, control statements, dispose files, and mass storage.

Option	Description

Example:

RESOURCE USAGE ALLOWED.

SECONDS	64
JOB STEP SRU	128
ACCOUNT BLK SRU	640
DAYFILE MESSAGES	462
CONTROL STATMTS	458
DISPOSE FILES	4
MASS STORAGE	12586

jobname — Last 3 characters of the name assigned by the system to a job initiated by the SUBMIT, ROUTE, or LDI statement. When this parameter is specified, the status of the job is returned. If JN (without =jobname) is specified, the status of all jobs associated with the current user number that are active in the system is returned. The user can obtain only the status of jobs submitted under the current user number.

lfn_1 — Local file name. When this parameter is specified, the status of the particular file is returned in the same manner as when the F option is specified.

lfn_2 — Name of alternate file to receive output. If omitted, the system assumes OUTPUT.

The third form of the statement (ENQUIRE.) defaults to the OP=A option. All OP= options (except S and T) are executed, and the information is printed on the OUTPUT file.

If the JN=option or FN=option is executed, the information is printed on the OUTPUT file only if it is the OUTPUT file for an interactive terminal. Otherwise, this information is written in the user's dayfile.

ENTER STATEMENT

The ENTER control statement enables the user to enter a series of control statements on one line. This is especially useful for time-sharing users operating in the batch subsystem.

The control statement format is:

ENTER./$statement_1$/$statement_2$/.../$statement_n$

/ — Delimiting character used to separate the individual control statements on one line. It can be any character not used within the control statements. It must immediately follow a period or right parenthesis.

$statement_i$ — Any NOS control statement for which the user is validated. Time-sharing commands for which there are no batch counterparts are not acceptable.

The system supplies a terminator (period or right parenthesis) if it is missing from any statement.

Example:

From a terminal, a user enters the batch subsystem and types in an ENTER statement on one line as follows:

```
BATCH
$RFL,0.
/ENTER.#SETCORE(0)#MAP(ON)#FTN(EL=F,I=ENTRFIL#LGO.#OVL.#DMP.#DMP
  (1000)
```

This is essentially the sequence of control statements in the job in section 12 used to illustrate the reading of CM dumps. However, instead of the FORTRAN program being in the INPUT file, it is on a permanent file called ENTRFIL. The printouts shown in the figures in section 12 are automatically listed at the terminal after the user presses carriage return at the end of the ENTER statement.

EXIT STATEMENT

The EXIT control statement indicates the position in the control statement record where processing will resume if an error is encountered or where to terminate normal control statement processing if an error is not encountered. For additional information, refer to the description of the NOEXIT and ONEXIT control statements later in this section and to the description of exit processing in section 5.

The control statement format is:

```
EXIT.
```

HTIME STATEMENT

The HTIME control statement issues a dayfile message giving the CYBER 170 model 176 accumulated clock cycle count for the job. A clock cycle on the model 176 is 27.5 nanoseconds. COMPASS instructions require a certain number of clock cycles to execute as described in the COMPASS reference manual. This control statement can be used for performance comparisons.

The control statement format is:

```
HTIME.
```

The resulting dayfile message has the following format. The cycle count is in kilocycle units.

```
HTIME nnnnnnnnnnnn.nnn KCYCLES.
```

An HTIME statement processed on a machine other than the model 176 produces the following dayfile message.

```
HTIME NOT AVAILABLE.
```

LDI STATEMENT

The LDI routine copies lfn to mass storage and submits the job(s) to the input queue with IDs to identify each job. The copy begins at the current position of the file pointer and continues until an EOI or double EOF is encountered. The jobs submitted are batch origin type jobs. LDI does no reformatting of the job file and therefore submit directives (/job, /NOSEQ, and so forth) are not allowed.

The control statement format is:

LDI(lfn, id, m)

lfn — Name of file containing the job(s) to be submitted; if lfn is omitted, LOAD is assumed.

id — Identification code (0 through 67_8 and 77_8); if omitted, 0 is assumed. If an id of 77_8 is assigned, the OUTPUT file is released at job completion.

m — Job names of jobs loaded are listed in the dayfile for the control point; if omitted, the list is suppressed.

The user can submit only the number of jobs for which he is validated (refer to the DB field description for the LIMITS control statement in this section). If this limit is exceeded, no further jobs are loaded, and the following message is issued to the dayfile.

TOO MANY DEFERRED BATCH JOBS.

If the submitted job contains an illegal USER statement, the job entering the LDI statement is aborted (no exit processing), and the following messages are issued to the dayfile.

ILLEGAL USER CARD.
SYSTEM ABORT.

In addition, the following message is issued to the account dayfile.

SIUN, usernum.

Terminal users are immediately logged off with no dayfile message. The security count for the user number that entered the LDI statement is decremented accordingly.

LENGTH STATEMENT

The LENGTH control statement gives the user the current status of one of his local files.

The control statement format is:

LENGTH(lfn)

lfn — Name of local file.

The information given for the local file includes its length in PRUs, type, and current status.

LIMITS STATEMENT

The LIMITS control statement directs the system to list validation information on file OUTPUT for the user named on the latest USER statement.

The control statement format is:

LIMITS.

Generally, validation limits are the internal system controls associated with each user number which govern his use of certain system resources. The listing provided describes both the resources available to the user and the extent to which they may be used. All numeric values listed are decimal unless the postradix B appears, signifying an octal value. The following information is listed.

Field	Description
AB†,††	Answerback identifier (1 to 10 alphanumeric characters) used for terminal identification.
MT	Maximum number of magnetic tape units the user is allowed to have assigned to his job concurrently.
RP	Maximum number of removable auxiliary devices the user is allowed to have assigned to his job concurrently.
TL	Maximum amount of central processor time (cumulative CPU time slices) in seconds allowed for each job step of the user's job. TL represents the actual time limit divided by 10_8.
CM	Maximum number of central memory words that the user is allowed to request. The value stored for CM represents the actual word limit divided by 100_8.
NF	Maximum number of files that the user is allowed to have assigned to a job concurrently.
DB	Maximum number of deferred batch jobs that the user can have in the system concurrently. If the user is validated for system privileges and DEBUG mode is set on the system display console or if the user is submitting jobs from system origin, this parameter is ignored. The user is allowed to submit as many jobs as desired.
FC	Maximum number of permanent files the user can have in the catalog.
CS	Maximum number of PRUs available to the user for indirect access files.
FS	Maximum number of PRUs available to the user for any one indirect access file.
PA†,††	Terminal parity (EVEN or ODD).
RO†,††	Number of rubout characters required for carriage return delay.
PX†,††	FULL or HALF duplex transmission mode.
TT†,††	Terminal type.
TC†	Character set to be used by time-sharing terminal.
IS†	Initial subsystem for time-sharing terminal.

† For further information about this field, refer to the IAF Reference Manual or Time-Sharing User's Reference Manual.

†† These fields are not used with network terminals.

Field	Description
MS	Maximum number of mass storage PRUs the user is allowed to additionally allocate via his job.
DF	Maximum number of CPU program messages that the user's job can issue to the system and/or job dayfiles.
CC	Maximum number of batch control statements processed for a user. (Time-sharing processed control statements are excluded.)
OF	Maximum number of print and punch files the user can dispose to output queues.
CP	Maximum number of cards that can be punched from a user's punch file.
LP	Maximum number of lines that can be printed from a user's print file.
EC	Maximum number of ECS memory words that the user is allowed to request.
SL	Maximum number of SRUs the user is allowed for a job.
CN	Charge number to which the user is assigned.
PN	Project number to which the user is assigned.
DS	Maximum number of PRUs available to the user for any one direct access permanent file.
AW	Access word; controls the user's access within the system according to the following options (assumed values are options 0, 2, and 3).

Option	Specifies
0	User can change his password.
1	User can use the privileged time-sharing commands.†
2	User is allowed to create direct access files.
3	User is allowed to create indirect access files.
4	User can have system origin (SYOT) capability for any job origin if the system console is in DEBUG mode. The user is allowed to assign a device by its EST ordinal although the system need not be in DEBUG mode to do so. The user is allowed to call the customer engineering PPU-based diagnostics if ENGINEERING mode (ENGR) is set at the system console.
5	User can access/create library files.
6	User can assign nonallocatable devices. A nonallocatable device is a magnetic tape unit, card reader, card punch, or line printer. Refer to the REQUEST statement in section 7 for further information.
7	User is allowed to access the system without supplying his assigned charge and project numbers.
8	User can define, save, and replace files on auxiliary devices.
9	User can access special transaction functions for library updates and batch transaction processing.

† For further information about privileged time-sharing commands, refer to the NOS Operator's Guide.

Option	Specifies
10	Allows no terminal timeout.
11	Allows use of the system control point (SCP) facility.
12	User has special accounting privileges. †
13	Allows BATCHIO subsystem privileges. ††
14	Allows use of the PROTECT statement.
15-23	Reserved.
24-35	Used by Control Data for application validation. †††
36-47	Available for user application validation.
48-59	Reserved.

The numerical value listed for AW is an octal representation of the bit settings for the above options. Thus bit 0 is option 0, bit 1 is option 1, and so forth. The rightmost octal number can designate any combination of options 0, 1, and 2; the next octal number to the left can designate any combination of options 3, 4, and 5; and so on. For example, if the access word were:

AW=0000000000100000215

the user would be validated for options 0, 2, 3, 7, and 24.

If any parameters are included on the LIMITS statement, the system issues the following message to the user's dayfile.

ERROR IN LIMITS ARGUMENTS.

MFL STATEMENT

The MFL control statement resets the maximum field length for each subsequent job step. The control statement format is:

MFL(nnnnnn, mmmm)

or

MFL(CM=nnnnnn, EC=mmmm)

nnnnnn	Maximum central memory field length (octal is assumed unless decimal is specified by a D suffix or use of the digits 8 or 9).
mmmm	Maximum extended core storage (ECS) field length. The value of mmmm is the actual extended core field length divided by 1000_8.

The parameters may be specified positionally, by keyword, or intermixed positionally and by keyword. If intermixed, the positional parameters are evaluated according to their position among all the parameters.

† Refer to the System Maintenance Reference Manual for a description of special user's accounting privileges.

†† Currently this bit allows the user to use the V carriage control character (refer to appendix I).

††† These options are described in the System Maintenance Reference Manual.

The parameter nnnnnn sets an upper boundary for the field length of subsequent job steps. The value cannot exceed the maximum field length for the job nor can it be less than 1500, the field length required for the utility (CONTROL) that processes MFL. Likewise, the parameter mmmm sets an upper boundary for the ECS field length of subsequent job steps and cannot exceed the maximum field length for the job. If the value 0 (zero) is entered for CM or ECS field length, the MFL is set to the maximum field length for the entire job.

The MFL control statement clears any initial running field length previously established with the RFL control statement or the SETRFL macro and allows the system to determine the field length for each succeeding job step. The system continues to determine field lengths until another RFL control statement or SETRFL macro is encountered.

If the field length requested is greater than 377777_8 for CM, or 7777_8 for EC, the following error message is issued.

CM OR EC REQUEST EXCEEDS MAXIMUM.

MODE STATEMENT

The MODE statement defines the error conditions that cause the system to exit from normal processing. When the specified error occurs, the system sets the appropriate error flag and exits from normal processing to perform any error processing required. If an error occurs for which the exit mode is not selected, the system notes the error, skips the operation that is causing the error, and continues normal processing.

The control statement format is:

MODE(m, n)

m	CPU program error exit mode ($0 \leq m \leq 7$)
n	CPU hardware error exit mode ($0 \leq n \leq 7$). Included for compatibility with earlier versions of NOS. The system now forces n=7 regardless of the value specified on the control statement.

The following values can be supplied for m.

m	CPU Program Error Exit Mode
0	Disable program exit mode; no selection made.
1	Address out of range because: • Attempt was made to reference CM or ECS outside established limits, or • Attempt was made to reference last 60-bit word (word 7) in relative address FL of ECS.
2	Operand out of range; floating-point arithmetic unit received an infinite operand.
3	Address or operand out of range.
4	Indefinite operand; floating-point arithmetic unit received an indefinite operand.
5	Indefinite operand or address out of range.
6	Indefinite operand or operand out of range.
7	Indefinite operand, operand out of range, or address out of range. If no mode is selected, the system assumes m=7.

If exit mode 3, 5, 6, or 7 is specified, a combination of exit modes 1, 2, and 4 is actually selected. For example, if exit mode 5 is specified, an error exit occurs for either a mode 1 or mode 4 error condition. Refer to Error Control in section 3 and to the CYBER 170, CYBER 70, and 6000 Series Computer Systems Reference Manuals for further information about the processing of mode errors.

NOEXIT STATEMENT

The NOEXIT control statement suppresses the transfer of control to the statement following the next EXIT statement if an error occurs.

The control statement format is:

```
NOEXIT.
```

If a NOEXIT statement has appeared in the control statement record and an error occurs, processing continues with the next control statement, if possible (that is, if the error does not cause the job to unconditionally abort). Refer to the description of exit processing in section 5 for further information.

NORERUN STATEMENT

The NORERUN control statement allows a user to clear job rerun status.

The control statement format is:

```
NORERUN.
```

If the NORERUN statement has been issued, the job may not be rerun. This may be desirable to prevent updating of an important data base when the job would otherwise be rerun.

This statement is ignored from a time-sharing origin job.

NOTE STATEMENT

The NOTE control statement enables the user to create a file containing lines of data entered as a character string on the same line as the control statement.

The control statement format is:

NOTE(lfn, NR)/$line_1$/$line_2$/.../$line_n$

lfn	Name of the file which contains the specified lines. Default is OUTPUT.
NR	Inhibits rewind of lfn. Default is to rewind the file at the beginning and end of NOTE execution.
/	Delimiting character used to separate the individual entries that become lines in the file. It can be any character. It must immediately follow a period or right parenthesis.
$line_i$	A character string which constitutes one line of data in lfn.

If a file contains more lines than can be entered with a single NOTE statement, a series of NOTE statements, each with an NR, can be used. This series should be followed with a PACK statement since each NOTE statement writes an EOF.

Example:

The following sequence of statements creates a procedure file (PFILE) that can insert an input record after any record in an existing master file (LISTFIL).

```
NOTE(PFILE,NR)*.PROC,INSERT,N.*GET(LISTFIL)*COPYBR(LISTFIL,NEWLIST,N)
NOTE(PFILE,NR)*COPYBR(INPUT,NEWLIST)*COPYEI(LISTFIL,NEWLIST)
NOTE(PFILE,NR)*REPLACE(NEWLIST=LISTFIL)
PACK(PFILE)
SAVE(PFILE)
```

To insert an input record after the second record in LISTFIL, the user includes the following CCL statement in the job that contains the new input record.

```
BEGIN,INSERT,PFILE,2.
```

OFFSW STATEMENT

The OFFSW control statement clears the pseudo-sense switches for reference by the user's program.

The control statement format is:

OFFSW($s_1, s_2, \ldots, s_n$)

s_i	Sense switch to be cleared; $1 \leq s_i \leq 6$. If $s_i=0$ is specified, all sense switches are cleared.

Refer to the description of the ONSW statement for further information on sense switch settings.

ONEXIT STATEMENT

The ONEXIT control statement causes the transfer of control to the statement following the next EXIT statement if an error occurs.

The control statement format is:

ONEXIT.

The ONEXIT statement reverses the effect of a NOEXIT statement. If an error occurs in processing the statement following ONEXIT, control transfers to the statement following the next EXIT statement. Refer to the description of exit processing in section 5 for further information.

ONSW STATEMENT

The ONSW control statement sets the pseudo-sense switches for reference by the user's program.

The control statement format is:

ONSW($s_1, s_2, \ldots, s_n$)

s_i	Sense switch to be set; $1 \leq s_i \leq 6$. If $s_i=0$ is specified, all sense switches are set.

The system stores the sense switch settings in the user's control point area and copies them to RA at the beginning of each job step for use by the central program. The sense switch field in the control point area and the one in RA are updated separately.

PASSWOR STATEMENT

The PASSWOR control statement is used to change the user's password.

The control statement format is:

PASSWOR(oldpswd, newpswd)

oldpswd	Old password
newpswd	New password

The new password must be the minimum length required by the installation. The default minimum is 4 characters.

For added security, the user may issue the PASSWOR statement without parameters. In this case, the system reads the parameters from a record in the INPUT file. This record must be a single line with the following format.

oldpswd, newpswd

The user's password is changed from oldpswd to newpswd. The user can change his password only if access option 1 is set (refer to the LIMITS control statement in this section). If option 1 is not set and the user submits a PASSWOR statement, the system issues the following message to his dayfile.

ILLEGAL CONTROL CARD.

If the control statement parameters are in error, the system issues the following message.

ERROR IN PASSWOR ARGUMENTS.

If the installation is currently updating the validation file or another user is modifying his password, a nontime-sharing origin job is rolled out until the validation file is available. A time-sharing origin PASSWOR command is aborted with the message:

MODVAL ABORTED.

If this situation is encountered, the time-sharing user should be able to retry his password change within a short time.

PROTECT STATEMENT

The PROTECT statement is used to activate or deactivate preservation of a user's ECS field length between job steps.

The control statement format is:

PROTECT ($\left\{\begin{matrix}\text{ON}\\\text{OFF}\end{matrix}\right\}$)

or

PROTECT (EC= $\left\{\begin{matrix}\text{ON}\\\text{OFF}\end{matrix}\right\}$)

The parameter is activated by specifying ON and deactivated by specifying OFF. ECS preservation is initially OFF.

Ordinarily, the ECS field length of a job is zeroed at the completion of a job step. With EC=ON, the ECS field length is preserved between job steps.

The PROTECT statement is available to the user only if option 15 of his access word is set (refer to the LIMITS control statement in this section). If option 15 is not set and the user submits a PROTECT statement, the system issues the following message to his dayfile.

CPM - ILLEGAL USER ACCESS.

If no parameters are specified, an illegal keyword is used, or any parameter other than ON or OFF is entered, the system issues the following message.

ERROR IN CONTROL ARGUMENTS.

RERUN STATEMENT

The RERUN control statement allows a user to set job rerun status.

The control statement format is:

RERUN.

If the RERUN statement has been issued, the job may be rerun. This statement is ignored from a time-sharing origin job.

RESOURC STATEMENT

The RESOURC control statement is required in any job that uses more than one tape or pack concurrently; it prevents deadlocks with other jobs which may need the same resources.

The control statement format is:

RESOURC($rt_1=u_1, rt_2=u_2, \ldots, rt_n=u_n$)

rt_i Resource type:

LO	7-track magnetic tape unit
HI	7-track magnetic tape unit
HY	7-track magnetic tape unit
HD	800 cpi, 9-track magnetic tape unit
PE	1600 cpi, 9-track magnetic tape unit
GE	6250 cpi, 9-track magnetic tape unit
MT†	7-track magnetic tape unit
NT†	9-track magnetic tape unit (800/1600 cpi)
DIi	844-21 Disk Storage Subsystem ($1 \leq i \leq 8$)
DJi	844-41/44 Disk Storage Subsystem ($1 \leq i \leq 8$)
DKi	844-21 Disk Storage Subsystem (full track) ($1 \leq i \leq 8$)
DLi	844-41/44 Disk Storage Subsystem (full track) ($1 \leq i \leq 8$)
DMi	885 Disk Storage Subsystem (half track) ($1 \leq i \leq 3$)
DQi	885 Disk Storage Subsystem (full track) ($1 \leq i \leq 3$)

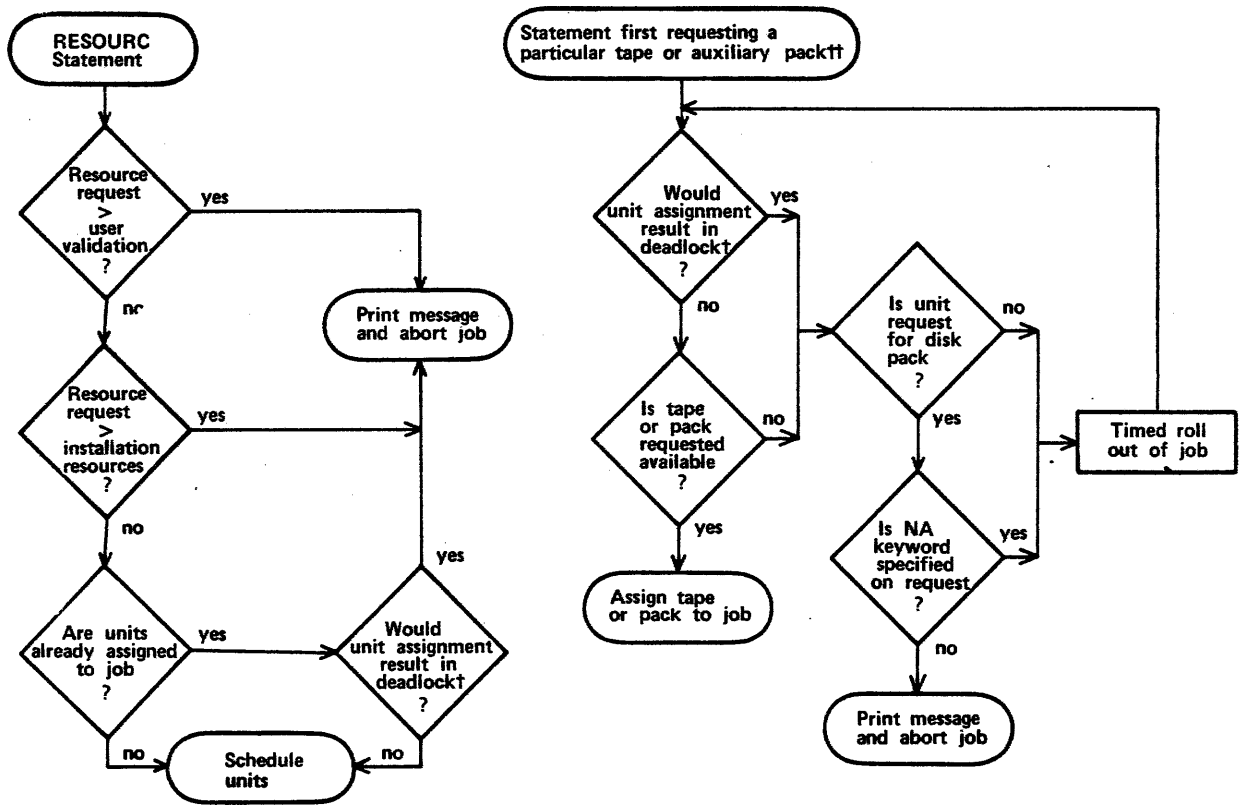
u_i Maximum number of units of resource type rt_i this job will use concurrently; any $rt_i=u_i$ entry can be changed on subsequent RESOURC control statements. (Refer to Altering Resource Requirements.) An rt=0 entry can be entered to clear a resource type that is no longer required.

† Retained for compatibility with NOS 1.2. Refer to restrictions described under Tape Units in this section.

DEADLOCK PREVENTION

The system manages the use of tape units and disk packs so as to prevent deadlocks from occurring. A deadlock means that the system, by assigning a tape unit or pack to one job, prevents another job with currently assigned resources from completing. For example, an installation with two tape units is processing jobs A and B. Each job needs both units during some phase of processing. Job A is assigned unit 1. If job B were assigned unit 2, neither A nor B could complete until the other job relinquishes its assigned unit.

To prevent deadlocks from occurring, the system requires that a RESOURC control statement be included in any job that uses more than one tape or disk pack concurrently. Thus, in the previous example, a RESOURC statement is required in both jobs. The information supplied by the statements enables the system to anticipate the deadlock situation and roll out job B until job A no longer needs both units. When a job that includes a RESOURC statement is submitted, the system first checks if the specified number of units exceeds the number of units for which the user is validated† or the number of units available at the installation. If either of these situations occurs, the system issues an error message to the user's dayfile and aborts the job. (Refer to figure 1-6-1.)



†Refer to description of Resource Overcommitment.
††The statements are described in sections 8 and 10.

Figure 1-6-1. Resource Commitment Processing (Simplified)

†For jobs that use only one tape or pack at a time and do not contain a RESOURC statement, the system checks validation limits when the request is made.

When the job requests a tape or pack,† the system compares the number of units that jobs being processed have scheduled via RESOURC statements with the number of units actually assigned. If it determines that the assignment would cause a deadlock (refer to Resource Overcommitment), it rolls out the job until a deadlock would not occur. If the assignment would not cause a deadlock, the system searches for the requested tape or pack. If found, it is assigned to the requesting job. If the pack is not found and the NA keyword was included in the request or if the tape is not found, the requesting job is rolled out until the operator makes the pack or tape available.

SINGLE RESOURCE USE

A job that uses only one tape or disk pack concurrently does not need to specify resource demand with a RESOURC statement. However, before assigning the same or a different type of resource, the current single resource (tape or disk pack) must be returned with either the RETURN or UNLOAD control statement. To allow more flexible resource handling, both the RETURN and UNLOAD functions decrement the default resource demand count from one to zero for jobs requiring only one tape or disk pack concurrently. For those jobs requiring more than one tape or disk pack concurrently (as specified by the RESOURC statement), UNLOAD does not decrement the resource demand count; RETURN decrements the resource demand count only when all concurrent resource demands have been satisfied.

TAPE UNITS

Density resource identifiers (HD, PE, GE) should be used to indicate 9-track magnetic tape unit demand. The system supports 9-track drives with alternate densities and needs this information to prevent deadlocks and overcommitments. The 679-2/3/4 tape units are capable of processing both 800-cpi and 1600-cpi 9-track tapes; the 679-5/6/7 tape units handle both 1600-cpi and 6250-cpi 9-track tapes. An 800-cpi 9-track tape cannot be processed on a 1600/6250-cpi unit, and 6250-cpi 9-track tape cannot be processed on an 800/1600-cpi unit. The NT resource identifier, retained for compatibility, can be used only to allocate 800/1600-cpi 9-track units†† and cannot be specified concurrently in the same job with HD, PE, and GE resource demands. Default 9-track resource allocation is by density.

Examples:

An installation has the following tape drive resources:

- Two 679-4 9-track tape drives (800/1600-cpi densities)
- Two 679-7 9-track tape drives (1600/6250-cpi densities)

† Refer to Permanent File Control Statements in section 8 for a description of disk pack requests and to Tape Management Control Statements in section 10 for a description of tape requests.

†† NT resource demand cannot exceed the number of 800/1600-cpi 9-track drives at the installation. However, at tape assignment time, a 1600-cpi tape mounted on a 1600/6250-cpi unit is accepted for NT resource demand if it does not cause overcommitment (potential deadlock).

1. If a job makes a tape unit resource request with

   RESOURC(NT=3)

   the job is aborted with the message

   INSUFFICIENT RESOURCES ON SYSTEM

   because only two units (the 679-4s) meet the NT specification.

2. If a job makes a tape unit resource request with

   RESOURC(NT=1,PE=1)

   the job is aborted with the message

   CONFLICTING RESOURCE TYPES.

   because the NT specification cannot be used with a density specification (HD/PE/GE).

3. If a job contains the following control statements

   ```
   LABEL(TAPE,NT,D=PE,VSN=TAPE1)
   RESOURC(NT=2)
   ```

   the job is aborted with the message

   CONFLICTING RESOURCE TYPES.

   because the LABEL statement requested a tape unit by density (the default); therefore, later statements cannot schedule tape units using the NT specification.

Density identifiers are provided for 7-track tape units even though these units do not have alternate densities. This is done for consistency of format. The LO, HI, HY, and MT resource identifiers are all equivalent, and the last specification of any one of these is the 7-track tape unit demand for the job. For example, the resource request RESOURC(HI=1, HY=2) results in two 7-track tape resources being allocated for the job.

RESOURCE OVERCOMMITMENT

Under certain conditions, the system overcommits resources, provided all jobs with currently assigned resources can complete. For example, an installation with three tape units is processing jobs A and B. Included in each job is a RESOURC statement scheduling two units. Job A requests its first tape. It is assigned the tape (unit 1) because there are enough units available for job A to complete. Job B requests its first tape. It is assigned the tape (unit 2) because either A or B can complete if assigned the last unit, and when the job that is assigned the last unit completes, the other can then use that unit and also complete. Job B then requests and is assigned its second tape (unit 3). It completes its operations (that is, terminates or returns the files on the tape) and makes the unit available for job A to complete.

NOTE

In a multimainframe environment, only the configuration of the machine on which the job is processed is considered in the overcommitment algorithm.

ALTERING RESOURCE REQUIREMENTS

The system manages its resources by keeping totals of the number of units of each device type scheduled and assigned to jobs. The number of units scheduled and the number of units assigned to a job can vary during job processing.

To change the number of units of a device type scheduled for this job, the user can issue another RESOURC statement. When decreasing the number of units scheduled for the job via a RESOURC statement, the total resulting scheduled units must not be less than the number of units currently assigned to the job. If the resulting total would be less than the number currently assigned, the system aborts the job with an error message.

If the job has tape and/or removable pack units assigned to it when it attempts to increase its resource demands, the system determines if the request would cause a deadlock. If it would, it aborts the job with an error message.

NOTE

It is recommended that the user always return all units assigned to his job before issuing another RESOURC statement to increase resource demands. This action prevents a possible deadlock condition resulting in job abort.

The scheduled units can also be decreased by a RETURN statement if the job, at a previous time, concurrently used its maximum scheduled units (refer to the description of the RETURN statement in section 7).

Example:

The second RESOURC statement increases the number of scheduled disk drives and decreases the number of scheduled tape units.

```
SAMSJOB(CM50000,T40)
USER(SJGREEN,WGT,ALTFAM)
CHARGE(D593,75)
RESOURC(HD=2)
LABEL(X,D=HD,VSN=TAPE1)
LABEL(Y,D=HD,VSN=TAPE2)
RETURN(X,Y)
RESOURC(DI1=2,HD=1)
    .
    .
    .
-EOI-
```

UNIT ASSIGNMENT

The method of assigning units depends on the resource type. For example, all tapes and all private disk packs not accessible by alternate users can only be assigned to one job at a time. All public packs and those private packs accessible by alternate users are shareable, and therefore, can be assigned to several jobs at the same time.

On indirect access file requests, the pack is charged to the job in fulfilling its resource demand only if the request causes the pack to be mounted. For direct access file requests, the pack is charged to the job when the first ATTACH of a direct access file is made.

A unit is assigned to a job until the job terminates or all direct access files residing on the unit that are assigned to the job are returned. At this point, a tape or a nonsharable pack can be dismounted. A sharable pack, however, can be dismounted only when there are no files residing on the unit that are assigned to any of the jobs sharing the pack.

NOTE

In GET requests for indirect access files, a pack is assigned to a job only as long as the pack is actually being used (that is, until the system retrieves the local copy of the file). Therefore, during a series of GET requests, the operator may determine that the pack is not being used and dismount it. If the user has a direct access file on the pack, he can avoid this situation by attaching the direct access file before issuing the GET requests.

A single job cannot have more than 36 removable pack devices attached to the job concurrently.

RFL STATEMENT

The RFL control statement sets the initial running field length for each subsequent job step when neither the routine for processing that step nor a loader table specifies a field length (refer to Field Length Control in section 3).

The control statement format is:

RFL(nnnnnn, mmmm)

or

RFL(CM=nnnnnn, EC=mmmm)

nnnnnn — Central memory field length (octal is assumed unless decimal is specified by a D suffix or use of the digits 8 or 9). The value is rounded up to the nearest multiple of 100_8.

Specifying nnnnnn as 0 removes the effect of the previous RFL statement and returns the setting of the field length to system control.

mmmm — ECS field length in octal. The value of mmmm is the actual ECS divided by 1000_8.

The parameters may be specified positionally, by keyword, or intermixed positionally and by keyword. If intermixed, the positional parameters are evaluated according to their position among all the parameters.

The values of nnnnnn or mmmm cannot exceed the values specified on the last MFL control statement or the maximum allowed for the job.

Prior to the appearance of the RFL control statement (or SETRFL macro), the system determines the field length for each job step, provided no field length is specified by a system routine or loader table (refer to Field Length Control in section 3).

If the field length requested is greater than 377777_8 for CM or 7777_8 for EC, the following error message is issued.

CM OR EC REQUEST EXCEEDS MAXIMUM.

ROLLOUT STATEMENT

The ROLLOUT control statement suspends job execution and places the job in the rollout queue. This releases the control point, central memory, and ECS assigned to the job. The user can specify a time period that must elapse before the job is returned. Otherwise, the job scheduler usually returns the job to execution when its priority is the highest of the jobs in the rollout queue (refer to Rollout Control in section 3).

The control statement format is:

ROLLOUT(t)

t	Optional time delay measured in job scheduler delay intervals. The delay interval length is set by the installation; the default value is 1 second. Legal values for t range from 0 to 262 080 (777700_8) intervals. Although the default base is decimal, octal values can be specified by a B suffix. Specifying a value containing an 8 or 9 and a B suffix is illegal.

RTIME STATEMENT

The RTIME control statement requests that the time be read from the real-time clock and issued to the dayfile (in seconds). This is the accumulated time since the last system deadstart.

The control statement format is:

RTIME.

The dayfile message format is:

RTIME nnnnnn.nnn SECS.

SETASL STATEMENT

The SETASL control statement sets the system resource unit (SRU) limit for an account block. An account block is the job step sequence whose execution is charged to an account (refer to SRU Limit Control in section 3). The account is specified by the charge and project numbers on a CHARGE statement, or if no CHARGE statement is required, by the user number on the USER statement. Each user number and each account has an SRU validation limit (refer to the LIMITS and ENQUIRE statements). Except for time-sharing jobs, the default account block SRU limit is the smaller of the user number and the account validation limits. For time-sharing jobs, the default limit is 64 SRUs.

The control statement format is:

SETASL(s)

s — Maximum number of SRUs allowed for account block execution. Although the default base is decimal, octal values can be specified by a B suffix on the value. Specifying a value with an 8 or 9 and a B suffix is illegal.

s must be greater than or equal to the current job step SRU limit, † and less than or equal to the user's and the account's validation limits. Exceptions to this rule are the asterisk (*) and values greater than 32 760 (77770_8). These exceptions set the account block SRU limit to the validation limit.

If the account block SRU limit is reached during account block execution, the system issues an error message and terminates the job (refer to Exit Processing in section 5).

SETCORE STATEMENT

The SETCORE control statement presets each word within the field length.

The control statement format is:

SETCORE(p)

or

SETCORE(-p)

p — Any of the following: (If a minus sign precedes the parameter p, the complement of p is set in core.)

p	Fill Characters
0	0
ZERO	Zeros (0)
INDEF	Indefinite (1777 0000 0000 0000 0000)
INF	Infinite (3777 0000 0000 0000 0000)

Each word within the field length is set to p. If p is omitted, the system assumes p=0.

To preset memory within a load sequence, the user issues a LDSET,PRESET control statement as described in the CYBER Loader Reference Manual.

† The job step SRU limit must be lowered in the job before the account block SRU limit is lowered. Refer to the SETJSL control statement in this section.

SETJSL STATEMENT

The SETJSL control statement sets the system resource unit (SRU) limit for each subsequent job step (refer to SRU Limit Control in section 3). Except for time-sharing jobs, the default job step SRU limit is the smaller of the user number and the account validation limits (refer to the LIMITS and ENQUIRE statements). For time-sharing jobs, the default job step limit is 64 SRUs. Time-sharing users can increment their job step SRU limit to complete job step execution (refer to the IAF Reference Manual or the Time-Sharing User's Reference Manual).

The control statement format is:

SETJSL(s)

s	Maximum number of SRUs allowed for job step execution. Although the default base is decimal, octal values can be specified by a B suffix on the value. Specifying a value with an 8 or 9 digit and a B suffix is illegal.
	s must be greater than 0 and less than or equal to the current account block SRU limit† and the user's and the account's SRU validation limits. Exceptions to this rule are the asterisk (*) and values greater than 32 760 (77770B). These values set the job step SRU limit at the validation limit if the account block SRU limit is set at the validation limit. †

The system issues an error message when the job step SRU limit is reached. A job step within a batch job is then terminated (refer to Exit Processing in section 5). In time-sharing jobs, the user can increment the SRU limit after receiving the SRU limit message (refer to the IAF Reference Manual or the Time-Sharing User's Reference Manual).

SETPR STATEMENT

The SETPR control statement allows the user to decrease the CPU priority of a job.

SETPR(p)

p	Priority, $1 \leq p \leq 70_8$; if p exceeds 70_8 or the maximum priority defined for the origin type of the job, it is reduced to that value.

Upon job initiation, a job is assigned the maximum priority allowed for its origin type. (The installation defines these priority values.) If a job's CPU priority is lower than that of other jobs, the job is assigned control of the CPU only when jobs of a higher priority do not need it.

† The account block SRU limit must be raised before the job step SRU limit can be raised. Refer to the SETASL control statement in this section.

SETTL STATEMENT

The SETTL control statement sets the CPU time limit for each subsequent job step. Each user number is validated for a maximum job step time limit (refer to the LIMITS and ENQUIRE control statements). For batch jobs, when the user does not specify a time limit, the system sets the limit at the user's maximum validation. For time-sharing jobs, the default time limits is 64 CPU seconds. Time-sharing jobs can increment their job step time limit to complete job step execution (refer to the IAF Reference Manual or the Time-Sharing User's Reference Manual).

The control statement format is:

SETTL(t)

t — Maximum number of CPU seconds allowed for job step execution. Although the default base is decimal, octal values can be specified by a B suffix on the value. Specifying a value with an 8 or 9 digit and a B suffix is illegal.

t must be greater than 0 and less than or equal to the user's validated time limit. Exceptions to this rule are the asterisk (*) and values greater than 32 760 (77770_8). These values set the job step time limit at the user's validated time limit.

The system issues an error message when the job step time limit is reached. A job step within a batch job is then terminated (refer to Exit Processing in section 5). In time-sharing jobs, the user can increment the time limit after receiving the time limit message.

STIME STATEMENT

The STIME control statement requests that the accumulated SRU value for the job be issued to the user's dayfile.

The control statement format is:

STIME.

The dayfile message format is:

STIME nnnnnn.nnn UNTS.

SUBMIT STATEMENT

The SUBMIT control statement places a user-supplied job file into the input queue as a separate job. SUBMIT can reformat the file according to directives within the file.

The control statement format is:

SUBMIT(lfn, q, NR)c

lfn — Name of the file to be submitted to the system for processing as a batch job.

q — Specifies disposition of job output files (OUTPUT, PUNCH, PUNCHB, and P8) as follows:

- B — Job output is disposed to local batch queue to be printed and/or punched at the central site (default value for nontime-sharing origin jobs).
- N — Job output is discarded at job termination (default value for time-sharing origin jobs).
- E — Job output is disposed to the remote batch queue for printing at a remote batch terminal.

NR — No rewind option; inhibits rewind of file specified by reformatting directive cREAD. If omitted, file specified by cREAD directive is automatically rewound.

c — Escape character used to identify reformatting directives in the file to be submitted (lfn). If omitted, the system assumes c=/.

The number of deferred batch (LDI, SUBMIT, and ROUTE) jobs that the user can have in the system concurrently depends on his validation (refer to the DB field of the LIMITS control statement in this section). If this limit is exceeded, an error message is issued to the dayfile, and the SUBMIT statement is not processed.

For SUBMIT to process reformatting directives, the first line of the submit file must be a cJOB directive. Each line preceded by an escape character is recognized as a reformatting directive. The escape character is specified on the SUBMIT statement (/ by default). Throughout this description, the letter c, preceding a directive, denotes the escape character. Reformatting directives may be interspersed throughout the submit file as long as transparent mode is not in effect. Transparent mode is selected by the cTRANS directive and requires that the user observe special rules when inserting subsequent directives into the file (refer to description of cTRANS and cNOTRANS directives).

The system does not process reformatting directives unless the first line of the submit file contains the cJOB directive. In addition, the first two statements following the cJOB directive (second and third statements of the submit file) must be a job and USER statement, respectively. All following information is determined by the user. Thus, the first three lines of a submit file to be reformatted should be:

```
ln1 cJOB
ln2 jobname,...
ln3 USER,...
```

ln1, ln2, and ln3 are optional line numbers.

The user can include line numbers on the submit file and specify which line numbers are to be removed during reformatting with the SEQ and NOSEQ directives. This is especially useful if the submit file contains a BASIC program where line numbers are a requirement of the language. If line numbers are included in a submit file, the file must begin with a cJOB directive.

The reformatting directives available are described as follows:

cJOB — Indicates that the submit file is to be reformatted and selects the following default reformatting directives. The default directives remain in effect until specified otherwise.

cNOTRANS	(disabled by cTRANS)
cSEQ	(disabled by cNOSEQ)
cPACK	(disabled by cNOPACK)

The cJOB directive must be the first line of the submit file. If omitted, the file is not reformatted. If line numbers are included in a submit file, the file must begin with a cJOB directive.

cEOR — Indicates that an end-of-record mark is to be placed at this point in the submit file during reformatting.

cEOF — Indicates that an end-of-file mark is to be placed at this point in the submit file during reformatting.

cSEQ — Indicates that the following lines are preceded by line numbers and requests that they be removed (default value).

cNOSEQ — Reverses the effect of the cSEQ directive. No attempt is made to remove leading line numbers from subsequent lines. This is especially useful when line numbers are required (such as in a BASIC program).

cPACK — Requests that all succeeding end-of-record and end-of-file marks be removed (default value). This directive applies only to internal EOR and EOF marks that currently exist. The cEOR and cEOF reformatting directives are not affected.

cNOPACK — Reverses the effect of the cPACK directive. Requests the system not to discard succeeding internal end-of-record and end-of-file marks that currently exist.

cTRANS — Requests transparent mode. In transparent mode, SUBMIT ignores reformatting directives until an EOR or EOF mark is encountered. The EOR or EOF mark cannot be a mark to be created by a cEOR or cEOF directive. SUBMIT performs the following procedure for transparent mode processing.

1. Read cTRANS directive.
2. Check if the next line is a reformatting directive. If it is not, skip steps 3 and 4.
3. Process reformatting directive. If it is a cNOTRANS directive, select nontransparent mode and end transparent mode processing.
4. Return to step 2.
5. Select transparent mode and read lines until an internal EOR or EOF mark is encountered.
6. If the cPACK directive is in effect, remove the EOR or EOF mark.
7. Return to step 2.

The cTRANS directive is typically used in conjunction with the cREAD directive. It allows the user to copy the contents of an existing file into the submit file at the location of the cREAD directive. Because the file is read in transparent mode, no check for reformatting directives is attempted until an internal EOR or EOF is encountered. The cREAD directive must follow the cTRANS directive and must be located before the first succeeding line that is not a reformatting directive. If not, transparent mode is selected before the cREAD directive is encountered and the cREAD is ignored.

The cSEQ or cNOSEQ directive in effect before transparent mode was selected has no effect upon the submit file or the file being read (cREAD) while transparent mode is in effect. However, the cPACK or cNOPACK directive in effect before transparent mode was selected remains in effect after it is selected.

cNOTRANS

Reverses the effect of the cTRANS directive and informs the system that the submit file is to be examined on a line-by-line basis. All directives encountered in the submit file while the cNOTRANS directive is in effect are processed. This directive is initially selected by default and remains in effect until a cTRANS directive is encountered in the submit file.

The user should be careful in placing this directive in the submit file. If transparent mode is selected, this directive can possibly be ignored unless it immediately follows either a cREAD directive or an internal EOR or EOF mark.

cREAD, lfn

Requests that the system read the contents of the specified file, lfn, and insert that file in place of the cREAD directive in the submit file, during reformatting. Reading terminates when an EOF or EOI is encountered on lfn. If the file to be read is not currently local to the job, the system automatically attempts a GET and then an ATTACH on the file. If lfn is not specified in the directive, TAPE1 is assumed. If the file specified cannot be found, the message

NO READ FILE - lfn.

is issued to the user's dayfile, and the job is terminated. If the read file is found to be busy (direct access files only), the message

READ FILE BUSY - lfn.

is issued to the user's dayfile, and the job is terminated. The file specified by lfn in the cREAD directive is automatically rewound before the read operation unless the NR parameter is specified on the SUBMIT control statement. In this case, the rewind directive must precede the cREAD directive in the submit file if it is desired to rewind file lfn before the read operation begins. The system returns all files specified in cREAD directives before completion of the job.

If the cPACK directive is in effect at the time of the read, all internal EOR marks are removed. If the cNOPACK directive is in effect, all internal EOR marks are read into the submit file in the proper position during reformatting.

Unless transparent mode is in effect when file lfn is read, each line of that file is also checked for a reformatting directive. Any directives contained in the file, except another cREAD, are processed. The cREAD directive cannot be nested. In addition, any directives in effect before the cREAD directive is processed remain in effect for the file being read, unless transparent mode is selected. Then, only the cPACK or cNOPACK directive remains in effect for the file being read. Moreover, only those directives that immediately follow an internal EOR in the file being read are processed.

If the file to be read is a binary file, it is recommended that the cTRANS directive be used to ensure that binary data is not mistaken for a reformatting directive. The cTRANS directive should immediately precede the cREAD directive in the submit file, if used.

cREWIND,lfn — Requests that the system rewind file lfn to the beginning-of-information (BOI). If lfn is not supplied, TAPE1 is assumed. This directive is required only if the NR parameter is included in the SUBMIT command. Otherwise, file lfn is automatically rewound.

This directive is used in conjunction with the cREAD directive. Thus, if it is desired to rewind a file before the read operation begins, this directive must precede the cREAD directive in the submit file.

c_1EC=c_2 — Indicates that the escape code character is to be changed from c_1 (current escape code) to c_2 (new escape code). The new escape code is used to recognize all subsequent reformatting directives until further change.

Input lines must not exceed 150 6-bit characters. SUBMIT processes the first 80 characters as the control statement. The remaining 70 characters are discarded and may contain a sequence number or comments. If a line exceeds 150 characters, the results are unpredictable.

If the submitted job contains an illegal USER statement, the job entering the SUBMIT statement is aborted (no exit processing). The following messages are issued to the dayfile.

```
ILLEGAL USER CARD.
SYSTEM ABORT.
```

The security count for the user number that entered the SUBMIT statement is decremented, and the following message is issued to the account dayfile.

```
SIUN,usernum.
```

Terminal users are immediately logged off and no message is issued. The system then begins the login sequence (for IAF users) if the security count is greater than zero. For further information concerning use of the SUBMIT statement from a time-sharing terminal, refer to the IAF Reference Manual or the Time-Sharing User's Reference Manual.

The user should consult his job's dayfile to determine the cause of any errors that occurred during job processing. The dayfile for the submitted job is disposed to the local batch queue or the remote batch queue according to the disposition parameter on the SUBMIT statement.

When a user submits a batch job image from a time-sharing terminal, all output is dropped (unless requested otherwise by the disposition parameter). This includes the dayfile output. Therefore, the time-sharing user should make provisions within his job to save the contents of the dayfile if a processing error occurs. This is done by including the following control statements at the end of the control statement record.

```
lnx     EXIT.
lny     DAYFILE(lfn)
lnz     REPLACE(lfn)
```

SUI STATEMENT

The SUI control statement allows a user to access a permanent file catalog without using the USER statement.

The control statement format is:

SUI(n)

n — User index desired; $0 \leq n \leq 377777_8$.

The SUI statement is useful if validation is not active. Only system origin jobs may issue this control statement. If the job is not of system origin, the following message is issued.

CPM-ILLEGAL REQUEST.

SUMMARY STATEMENT

The SUMMARY control statement gives information about the system to the user. Three forms of the command are allowed.

The control statement formats are:

SUMMARY(OP=$p_1p_2 \ldots p_n$, JN=jobname, FN=lfn_1, O=lfn_2)

or

SUMMARY($p_1p_2 \ldots p_n$)

or

SUMMARY.

The parameters and function of this control statement are identical with the ENQUIRE statement described in this section, except that the third form of the statement (SUMMARY.) defaults to the OP=R option.

SWITCH STATEMENT

The SWITCH control statement sets the pseudo-sense switches for reference by the user's program.

The control statement format is:

SWITCH($s_1, s_2, \ldots, s_n$)

s_i	Sense switch to be set; $1 \leq s_i \leq 6$. If $s_i=0$ is specified, all sense switches are set.

Refer to the description of the ONSW statement for further information on sense switch settings.

This control statement performs the same function as the ONSW control statement.

USECPU STATEMENT

The USECPU control statement specifies which central processor is to be used when more than one is available for processing.

The control statement format is:

USECPU(n)

n = 0	Either central processor is used.
n = 1	CPU 0 is used.
n = 2	CPU 1 is used.

The USECPU statement may be used only when the system is running on a CYBER 73-2x, 74-2x, 6500, 6700, or CYBER 174 system. On a 74-2x or 6700, CPU 0 is the parallel processor, and CPU 1 is the serial processor. On the other systems, both CPUs are serial processors. This statement is ignored on single CPU machines.

USER STATEMENT

The system uses the parameters on the USER control statement to determine if a legal user initiated the job, which resources he is validated to use, and the extent (limits) to which he may use those resources. Comment statements are not allowed between the job and USER statements. If this is attempted, the first comment statement is interpreted as an illegal USER statement, and the submitting job is aborted with appropriate messages to the dayfile. The submitted job is dropped.

The control statement format is:

USER(usernum, passwrd, familyname)

usernum	A 1- to 7-character alphanumeric user number.
passwrd	Alphanumeric password. Its maximum length is 7 characters; its minimum length is defined by the installation.
familyname	Optional parameter identifying the family† of permanent file devices that have been or may be transferred from the user's normal system to a backup system.

This statement defines controls and validation limits for the job and defines the user's permanent file base. An installation may operate with secondary USER statements either enabled or disabled. If enabled, the user may specify a different permanent file catalog during job processing by issuing another USER statement. However, the access limits for the user named in the first USER statement remain in effect for all subsequent USER statements (refer to the LIMITS control statement in this section for information concerning access limits). If secondary USER statements are disabled (default mode) and a secondary USER statement is issued, the job is aborted (no exit processing). The security count for the current user number is decremented accordingly, and the following messages are issued to the dayfile.

ILLEGAL USER CARD.
SYSTEM ABORT.

In addition, the following message is issued to the account dayfile.

SIUN, usernum.

The job is aborted, the security count is decremented, and the preceding messages are issued if a USER statement containing an invalid user number is detected at any time, regardless of whether secondary USER statements are enabled or disabled. In all cases, terminal users are immediately logged off with no dayfile message issued to the terminal.

If the security count for the user is exhausted, the system issues the following message.

ILLEGAL USER NUMBER - CONTACT SITE OPR.

When this occurs, the user number is denied all access to the system until the security count has been reset by the installation personnel.

The password is deleted from the USER control statement before this statement is issued to the dayfile.

† Refer to section 2 for a description of permanent file device families.

Normally, the familyname parameter need not be included on the USER statement. However, if the user makes a practice of specifying his family name each time he submits a job, he can be sure that his job will be processed even if his normal system is not available and his permanent file family is moved to a backup system. If, after the first USER statement, the user does not specify a familyname on the USER statement, his permanent file family remains the same. If the user specifies the 0 (zero) familyname, his permanent file family becomes the system default family.

Example:

An installation has two systems, A and B. System B provides backup service for system A. The system default family name for system A is AFAM, and the system default family name for system B is BFAM.

During normal operations, system A user CWJONES with password JPWD could enter either of the following USER statements.

```
USER(CWJONES,JPWD)
USER(CWJONES,JPWD,AFAM)
```

System B user JDSMITH with password SPWD could enter either of the following statements.

```
USER(JDSMITH,SPWD)
USER(JDSMITH,SPWD,BFAM)
```

If system A failed, user CWJONES would be required to enter

```
USER(CWJONES,JPWD,AFAM)
```

to identify his family of permanent file devices. User JDSMITH could enter either of the USER statements as before because the default family name would still be valid.

If the user attempts to access permanent files on a device not present in the alternate system, one of the following messages is issued to the user's dayfile.

DEVICE UNAVAILABLE, AT nnn.	This message is issued if the user's master device† was not transferred to the backup system.
DIRECT ACCESS DEVICE ERROR, AT nnn.	This message is issued if the user attempted to reference direct access files on a device (other than his master device) not present in the backup system. †

† Refer to section 2 for a description of permanent file device families.

FILE MANAGEMENT CONTROL STATEMENTS 7

The file management control statements enable the user to manipulate files assigned to his job. The control statements included in this category are:

ASSIGN	COPYSBF	PACK	SKIPFB
BKSP	COPYX	PRIMARY	SKIPR
CLEAR	DISPOSE	RENAME	SORT
COMMON	DOCMENT	REQUEST	TCOPY
CONVERT	EVICT	RESEQ	TDUMP
COPY	FCOPY	RETURN	UNLOAD
COPYBF	LIST80	REWIND	UNLOCK
COPYBR	LOCK	ROUTE	VERIFY
COPYCF	LO72	SETID	WRITEF
COPYCR	NEW	SKIPEI	WRITER
COPYEI	OUT	SKIPF	

The statements in this section allow the user to position his files, copy data from one file to another, specify method and format of input/output, sort his files, and add corrections. He can assign his files to a specific device type; change the file type, identification code, and write interlock status; and release them from job attachment. The user can also receive information about records in a file or documentation in a file containing COMPASS source code.

If an error is encountered in an operation on one file of a multiple file request, the operation is not performed on the following files. For example, if an error occurs in processing file B on the following control statement:

```
GET(A,B,C,D)
```

files C and D are not processed.

If a file is not specifically assigned through the use of an ASSIGN, LABEL, or REQUEST control statement, the system assigns the file to available mass storage.

ASSIGN STATEMENT

The ASSIGN control statement directs the system to assign a file to the specified device or device type. The following descriptions refer to devices other than magnetic tape. For use of the ASSIGN statement with magnetic tape, refer to section 10.

The control statement format is:

ASSIGN(nn, lfn, $\left\{\begin{matrix}CK\\CB\end{matrix}\right\}$)

nn — Device or device type to which the specified file is to be assigned; nn may be either the EST ordinal† of a peripheral device or the device type as defined as follows:

Type	Equipment
DE	Extended core storage
DI	844-21 Disk Storage Subsystem (half track)
DJ	844-4x Disk Storage Subsystem (half track)
DK	844-21 Disk Storage Subsystem (full track)
DL	844-4x Disk Storage Subsystem (full track)
DM	885 Disk Storage Subsystem (half track)
DP	Distributive data path to ECS
DQ	885 Disk Storage Subsystem (full track)
MS	Mass storage device
NE	Null equipment
TT	Time-sharing terminals††

lfn — Name of the file to be assigned to the specified equipment

† Contact installation personnel for a list of EST ordinals.
†† This device type applies only to time-sharing origin jobs.

Example 1:

ASSIGN(MS, OUTPUT)

This statement assigns file OUTPUT to mass storage. With this assignment, a time-sharing user causes output normally printed at his terminal to be written on a mass storage file instead. Here, output means information generated by a program during execution. Dayfile messages are still printed at the terminal. Once this assignment is made, output is written on the mass storage file OUTPUT until the file is returned or reassigned.

Example 2:

ASSIGN(TT, XYZ)

This statement assigns file XYZ to the user's time-sharing terminal. The assignment means that input that the system would have read from file XYZ is instead solicited by a prompt at the terminal and that output that the system would have written on file XYZ is instead displayed at the terminal.

Example 3:

ASSIGN(DI, ABC)

This statement assigns file ABC to an 844-21 Disk Drive, if one is available.

The ASSIGN statement can also be used to create or access existing 7- or 9-track unlabeled tapes. For a description of the statement as it applies to magnetic tape assignment, refer to Tape Management in section 10.

BKSP STATEMENT

The BKSP control statement directs the system to bypass a specified number of logical records in the reverse direction.

The control statement format is:

BKSP(lfn, n, m)

lfn	Name of the file to be backspaced.
n	Number of logical records (decimal) to backspace; if this parameter is omitted, the system assumes n=1.
m	File mode: C for coded, B for binary. If omitted, the system assumes the file is in binary mode.

The BKSP request can be issued at any point in a logical record. If, for example, FILE1 were positioned within the third record, a

BKSP(FILE1)

request would reposition FILE1 to the beginning of the third record. The system does not backspace past the beginning-of-information (BOI) or load point (tape file). However, EOF indicators are considered separate records and are included in the record count. An unrecognizable record count causes the message

ERROR IN FILE ARGUMENTS.

to be issued to the user's dayfile.

The BKSP statement has no effect on a primary file since that file is rewound before every operation.

CLEAR STATEMENT

The CLEAR control statement releases all files currently assigned to the job. The user can also specify files that are not to be released.

The control statement formats are:

CLEAR.

or

CLEAR(*, lfn_1, lfn_2, . . . , lfn_n)

The first format releases all files. The second format releases all files except those named. If no files are named, all files assigned to the job are released.

Refer to RETURN statement in this section for the operations performed on each file type.

A CLEAR control statement should not be used in a CCL procedure because it returns the CCL working files. Further processing of the procedure produces unpredictable results.

COMMON STATEMENT

The COMMON control statement creates or accesses a library type file (LIFT).

The control statement format is:

COMMON(lfn_1, lfn_2, . . . , lfn_n)

lfn Logical file name.

The user must be validated to access or create library files. The specified file must be a local mass storage file. If lfn is not local, a search is made for a library file by that name, and an error message is issued if the file is not found. If the operation completes successfully, the file is attached to the user's job as a library type file.

Before a local file can be made a library file, it must be locked. Refer to LOCK Statement in this section.

CONVERT STATEMENT

The CONVERT control statement converts records from one character set to another.

The control statement format is:

CONVERT($p_1, p_2, \ldots, p_i$)

p_i May be one of the following.

- $P=lfn_1$ — Input on file lfn_1; if omitted, file OLD is assumed.
- $N=lfn_2$ — Output on file lfn_2; if omitted, file NEW is assumed.
- $RS=n_1$ — Maximum record size in characters (decimal); $1 \leq n \leq 500$. If omitted, 300 is the assumed maximum record size. (Each character is 6 bits.)
- 64 — Convert from 63- to 64-character set; if omitted, no conversion takes place. The TS option must be specified if 64 is not.
- TS=t — Convert from old time-sharing 61-character set to new time-sharing 63-character set; t may be one of the following terminal types.

t	Terminal Type
TTY	ASCII code terminal with standard print.
COR	Correspondence code terminal with standard print.
CORAPL	Correspondence code terminal with APL print.
MEMAPL	Memorex 1240 (ASCII code) terminal with APL print.
BLKEDT	Block transmission (ASCII code) terminal with full display screen editing capability and standard print.
NAMIAF	Virtual network terminal. Same as TTY.

If t is omitted, it is assumed to be TTY. If TS is omitted, no time-sharing conversion takes place. The 64 option must be specified if TS is not.

- R — Rewind input and output files prior to processing. If omitted, no rewind takes place.

RC=n_2 — Convert n_2 decimal records. If n_2 is omitted, convert until an EOF is encountered. If RC is omitted, one record is assumed.

NM — Used in conjunction with TS parameter and specifies that conversion is to normal mode; if omitted, conversion is to ASCII mode. Note the effect of conversion on the following characters.

^ (circumflex) — If TS is specified, display code 70 (circumflex character) is converted to 76. If NM is omitted, conversion is to 7402 (ASCII mode).

: (colon) — If TS and 64 are specified, display code 63 (colon character) is converted to 00. If NM is omitted, conversion is to 7404 (ASCII mode).

The following lists legal conversion using the appropriate CONVERT parameter.

Type of Record	Legal Conversion Parameters
63-character set, nontime-sharing record	64
Old time-sharing record	TS or 64 and TS
New NORMAL time-sharing record (equivalent to BATCH character set)	64
New ASCII time-sharing record	None

COPY STATEMENT

The COPY control statement copies data from one file to another if the files are within the range of permissible formats listed in table 1-7-1.

TABLE 1-7-1. RANGE OF PERMISSIBLE FORMATS FOR THE COPY STATEMENT

			Output					
			Mass Storage or Terminal	Tape Formats				
				I	SI	S	L	F
Input	Mass Storage or Terminal		Yes	Yes	Yes	Yes	Yes	No
	Tape Formats	I	Yes	Yes	Yes	Yes	Yes	No
		SI	Yes	Yes	Yes	Yes	Yes	No
		S	Yes	Yes	Yes	Yes	Yes	No
		L	Yes	Yes	Yes	No	Yes	No
		F	Yes	Yes	Yes	No	No	Yes

The parameters can appear in order-dependent format, order-independent format, or a combination of both. The completely order-dependent format is:

COPY(lfn_1, lfn_2, x, c, tc, copycnt, bsize, charcnt, erlimit, $p_1p_2 \ldots p_n$, lfn_3)

The completely order-independent format is:

COPY(I=lfn_1, O=lfn_2, V=x, M=c, TC=tc, N=copycnt, BS=bsize, CC=charcnt, EL=erlimit, PO=$p_1p_2 \ldots p_n$, L=lfn_3)

If order-dependent and order-independent parameters are mixed in one COPY statement, the order-dependent parameters must appear in their proper position. All parameters are optional. However, the specification of certain parameters precludes the application of others. A nonapplicable parameter may be ignored or it may be illegal. This is stated in the individual descriptions of the parameters.

The parameters are defined as follows:

Parameter	Description	Default
I=lfn_1	Name of the file to copy from.	INPUT
O=lfn_2	Name of the file to copy to.	OUTPUT
V=x	If the x parameter (1 to 7 alphanumeric characters) is present, both files are rewound, copied, rewound, verified, and rewound. The x parameter must not be zero.	No verify
M=c	M=C1 Coded mode is set on input only. M=C2 Coded mode is set on output only. M=any other value (1 to 7 alphanumeric characters) Coded mode is set on both input and output. This parameter applies only to S and L format tapes. If coded mode is set on an SI tape, the system aborts the job. For other formats, the system ignores the mode setting.	Binary
TC=tc	Specifies the copy termination condition used in conjunction with N=copycnt. The termination condition can be specified as follows:	Copy to double EOF (TC=D or TC=EOD)

Parameter	Description	Default
	tc — Meaning	
	F or EOF — The N keyword specifies the number of files to copy.	
	I or EOI — Copy to the end of information. The N keyword is ignored.	
	D or EOD — The N keyword is the number of double EOFs to copy to. If N > 1 is specified together with this TC value, and verify is also selected, the files are verified only to the first empty file (COPY calls VERIFY with N=0 parameter).	
N=copycnt	Copy count used with the copy termination condition specified by the parameter TC.	1
BS=bsize	Maximum block size (in central memory words) which specifies S or L tape PRU size. This applies only when copying to or from S and L tapes. It cannot be specified with the CC parameter.	If CC is not specified, 1000B for S tape copy and 2000B for L tape copy.
CC=charcnt	Maximum number of characters in an S or L tape block. This parameter can be specified only when copying to or from S and L tapes. The PRU size and unused bit count are calculated from the character count. However, the unused bit count is used only when writing a full block to an S or L output tape during a copy from mass storage, I, or SI format tape. The charcnt value should be a multiple of 10. If it is not, the characters that exceed the charcnt value in the last word of the record are discarded when writing an S or L format tape. This parameter cannot be specified with the BS parameter.	Not used (the PRU size is specified by the BS parameter)
EL=erlimit	Error limit which specifies the number of nonfatal errors allowed before abort. This includes both parity errors and block-too-large errors which are returned by the tape subsystem after completing recovery procedures. If EL=U is specified, unlimited error processing is allowed. Error recovery is supported on mass storage and on all tape formats but is not supported on a terminal or on unit record equipment. In the latter cases, any error aborts the job.	Zero

Parameter	Description		Default
PO=$p_1p_2\ldots p_n$	One or more of the following processing options:		
	E	Input blocks with parity errors or block-too-large errors are processed (copied).	Error blocks are skipped.
	D	Any noise blocks generated by a copy from mass storage, I format tape, or SI format tape to an S or L format tape are deleted. This parameter cannot be specified on any other type of copy.	For S or L binary tapes, noise blocks are padded to noise size with binary zeros; for coded mode, they are padded with blanks.
	R	Allows record splitting during a copy from mass storage, I format, or SI format to S or L format tape. This parameter cannot be specified on any other type of copy.	Record splitting is not allowed.
	M	Copy files according to the copy termination condition specified by the keyword TC, eliminating each EOF on output. This option is primarily for use with labeled S and L output tapes since it eliminates the conflict of the double meaning of a tape mark on these formats (the tape mark on these formats serves as both an EOF and label group delimiter).	Copy files according to specification of the copy termination (TC), writing an EOF after each file on output.
L=lfn_3	Name of an alternate output file to receive parity error messages when extended error processing is in effect (nonzero EL specified), in which case, the file name lfn_3 must not be the same as lfn_1 or lfn_2.		OUTPUT

Example:

The following COPY statement combines order-dependent and order-independent parameters.

```
COPY(FILE1,FILE2,VERIFY,CODED,EOF,6,L=MYOUT,PO=E,EL=10)
```

FILE1 is the input file, and FILE2 the output file. Six coded files are copied and verified. Up to 10 nonfatal errors are allowed, and the bad data is copied with informative error messages written to the file MYOUT.

The copy statement begins a copy operation at the current position of both files unless the verify option is specified. If verify is specified, both files are rewound before the copy begins and rewound, verified, and rewound again after the copy is completed. (This verify may not be meaningful if the logical structure of the two files is incompatible.)

Copy Termination

Copying continues until the copy termination condition is met or EOI is encountered. The copy termination condition can be a file count, a double EOF count, or EOI. If the copy is terminated by a double EOF (for TC=EOD option), the second EOF is detected on lfn, but is not transferred to lfn_2. If $lfn_1=lfn_2$ the named file is read until the termination condition is satisfied or EOI is encountered.

If a copy specifies a file count, TC=EOF, and EOI is encountered on the input file before the file count is satisfied, an additional EOF is written on the output file only if data or records have been transferred since the previous EOF was written (or since the beginning of the copy if no EOFs have been encountered).

Block Sizes

Both L and F tapes may require additional field length to accommodate their maximum block size. The maximum block size for an L tape copy is specified either by the BS=keyword (or its default), or it is calculated from the CC=keyword. The maximum block size for an F tape is determined by the maximum frame or character count specified when the file was assigned. The more accurate the selection of these values which determine block size, the less are the requirements for field length, CPU time, and I/O time.

Processing Options

The PO=D option specifies noise block processing, and the PO=R option specifies record splitting for copies from mass storage, I format, or SI format to S or L format tapes. Due to the incompatibilities between the logical structure of the input and output files, records may be encountered on the input file that are too small or too large to be copied directly to the S or L output tape. If the output file block size is less than noise block size, it is deleted if PO=D is specified. If PO=D is not specified, the block size is rounded to the word multiple of noise size with binary zero fill for a binary S or L tape or with blank fill for a coded S or L tape. Empty records on the input file are skipped since they cannot exist on an S or L tape. If PO=R is specified and an input file record length exceeds the S or L tape maximum block size (the PRU size as specified by BS= or its default, or by CC=), it is split into multiple blocks. If PO=R is not specified and an input record length exceeds the S or L tape maximum block size, the job aborts with the message

RECORD TOO LARGE ON lfn.

The PO=M option makes it possible to copy a multifile file to a labeled S or L format tape without writing the EOF tape marks. This avoids the conflict of a tape mark serving the double purpose of defining an EOF and delimiting a label group on S and L format tapes. This is in keeping with the tendency in the computer industry to define a tape mark only as a label delimiter.

The EL and PO=E options provide extended error processing. If EL is set to a value greater than zero, a parity error or a block-too-large error on the input file generates the following message on the alternate output file.

PARITY/BLOCK TOO LARGE ERROR IN BLOCK n.

n is the decimal block count of the block in error.

COPYBF STATEMENT

The COPYBF control statement copies a specified number of binary files from one file to another.

NOTE

The COPYBF statement produces unpredictable results when copying S, L, and F format tapes. The COPY utility is recommended for copying tapes in these formats.

The control statement format is:

COPYBF(lfn_1, lfn_2, n, c)

lfn_1	Name of the file to copy from; if this parameter is omitted, file INPUT is assumed.
lfn_2	Name of the file to copy to; if this parameter is omitted, file OUTPUT is assumed.
n	Number of files (decimal) on lfn_1 to copy; if this parameter is omitted, n=1 is assumed.
c	If a fourth parameter (1 to 7 alphanumeric characters) is present, the copy to or from an S or L format tape is performed in coded rather than binary mode. If coded mode is set on an SI tape, the system aborts the job. For other formats, the system ignores the mode setting.

The copy begins at the current position of lfn_1. If lfn_1=lfn_2, the file is read until the file count is satisfied or EOI is encountered.

If EOI is encountered on lfn_1 before the file count is satisfied, an additional EOF is generated on lfn_2 only if data or records have been transferred since the previous EOF was written (or since the beginning of copy if no EOFs have been encountered).

COPYBR STATEMENT

The COPYBR control statement copies a specified number of binary records from one file to another.

NOTE

The COPYBR statement produces unpredictable results when copying S, L, or F format tapes. The COPY utility is recommended for copying tapes in these formats.

The control statement format is:

COPYBR(lfn_1, lfn_2, n, c)

lfn_1	Name of the file to copy from; if this parameter is omitted, file INPUT is assumed.
lfn_2	Name of the file to copy to; if this parameter is omitted, file OUTPUT is assumed.
n	Number of records (decimal) to copy; if this parameter is omitted, n=1 is assumed.
c	If a fourth parameter (1 to 7 alphanumeric characters) is present, the copy to or from an S or L format tape is performed in coded rather than binary mode. If coded mode is set on an SI tape, the system aborts the job. For other formats, the system ignores the mode setting.

The copy begins at the current position of lfn_1. EOF indicators are considered separate records and are included in the record count. If lfn_1=lfn_2, the file is read until the record count is satisfied or EOI is encountered.

If EOI is encountered on lfn_1 before the record count is satisfied, an additional EOR is written on lfn_2 only if data has been transferred since the previous EOR or EOF was written (or since the beginning of the copy if no EORs or EOFs have been encountered).

COPYCF STATEMENT

The COPYCF control statement copies a specified number of coded files from one file to another. A coded file is defined as a file containing lines of 150 characters or less, each terminated by a zero byte (12 zero bits in the lowest byte of a word).

NOTE

The COPYCF statement produces unpredictable results when copying S, L, or F format tapes. The COPY utility is recommended for copying tapes in these formats.

The COPYCF statement cannot copy SI format tapes. If coded mode is set for an SI tape, the system terminates the job. The TCOPY utility converts SI coded tape files.

The control statement format is:

COPYCF(lfn_1, lfn_2, n, fchar, lchar)

lfn_1	Name of the file to copy from; if this parameter is omitted, file INPUT is assumed.
lfn_2	Name of the file to copy to; if this parameter is omitted, file OUTPUT is assumed.
n	Number of files (decimal) to copy; if this parameter is omitted, n=1 is assumed.
fchar	First 6-bit character position of each line to copy; if this parameter is omitted, fchar=1 is assumed.
lchar	Last 6-bit character position of each line to copy; if this parameter is omitted, lchar=136 is assumed.

The copy begins at the current position of lfn_1. If lfn_1=lfn_2, the file is read until the file count is satisfied or EOI is encountered. If EOI is encountered before the file count is satisfied, an EOF is written on lfn_2, and the operation terminates. If a line is encountered that has more than lchars, the excess characters are truncated.

COPYCF writes lines with an even number of characters. If an input line has an odd character count and the last character is a blank not immediately preceded by a colon, the last character is removed. If an input line has an odd character count and the last character is not a blank or is a blank immediately preceded by a colon, an additional trailing blank is appended.

If lchar is less than fchar, lchar is greater than 150, or either fchar or lchar is unrecognizable, the following error message is issued to the user's dayfile.

ILLEGAL CHARACTER NUMBER.

If COPYCF is attempted on a line longer than 150 (6-bit) characters, the following message is issued.

NO LINE TERMINATOR.

COPYCR STATEMENT

The COPYCR control statement copies a specified number of coded records from one file to another. A coded record contains lines of 150 characters or less, each terminated by a zero byte (12 zero bits in the lowest byte of a word).

NOTE

The COPYCR statement produces unpredictable results when copying S, L, or F format tapes. The COPY utility is recommended for copying tapes in these formats.

The COPYCR statement cannot copy SI format tapes. If coded mode is set for an SI tape, the system terminates the job. The TCOPY utility converts SI coded tape files.

The control statement format is:

COPYCR(lfn_1, lfn_2, n,fchar, lchar)

lfn_1	Name of the file to copy from; if this parameter is omitted, file INPUT is assumed.
lfn_2	Name of the file to copy to; if this parameter is omitted, file OUTPUT is assumed.
n	Number of records (decimal) to copy; if this parameter is omitted, n=1 is assumed.
fchar	First 6-bit character position of each line to copy; if this parameter is omitted, fchar=1 is assumed.
lchar	Last 6-bit character position of each line to copy; if this parameter is omitted, lchar=136 is assumed.

The copy begins at the current position of lfn_1. If lfn_1=lfn_2, the file is read until the record count is satisfied or EOI is encountered. EOF indicators are considered separate records and are included in the record count. If the EOI is encountered before the record count is satisfied, an EOF is written on lfn_2, and the operation terminates. COPYCR is processed in exactly the same manner as the COPYCF control statement except that n specifies the number of records rather than the number of files.

If COPYCR is attempted on a line longer than 150 (6-bit) characters, the following message is issued.

NO LINE TERMINATOR.

COPYEI STATEMENT

The COPYEI control statement copies one file to another.

NOTE

The COPYEI statement produces unpredictable results when copying S, L, or F format tapes. The COPY utility is recommended for copying tapes in these formats.

The control statement format is:

COPYEI(lfn_1, lfn_2, x, c)

lfn_1	Name of the file to copy from; if this parameter is omitted, file INPUT is assumed.
lfn_2	Name of the file to copy to; if this parameter is omitted, file OUTPUT is assumed.
x	If a third parameter (1 to 7 alphanumeric characters) is present, both files are rewound before the copy, and rewound, verified, and rewound again after the copy is complete.
c	If a fourth parameter (1 to 7 alphanumeric characters) is present, the copy to or from an S or L format tape is performed in coded rather than binary mode. If coded mode is set on an SI tape, the system aborts the job. For other formats, the system ignores the mode setting.

The copy begins at the current position of lfn_1 and continues until the EOI is encountered. The EOI is not defined for certain tape formats (refer to table 1-2-1).

If lfn_1=lfn_2, the file is read until EOI is encountered.

COPYSBF STATEMENT

The COPYSBF control statement enables the user to copy a file where the first character of each line is not a printer control character and is to be printed.

NOTE

The COPYSBF statement produces unpredictable results when copying S, L, or F format tapes. The COPY utility is recommended for copying tapes in these formats.

The COPYSBF statement cannot copy SI format tapes. If coded mode is set for an SI tape, the system terminates the job. The TCOPY utility converts SI coded tape files.

The control statement format is:

COPYSBF(lfn_1, lfn_2, n)

lfn_1	Name of the file to copy from; if this parameter is omitted, file INPUT is assumed.
lfn_2	Name of the file to copy to; if this parameter is omitted, file OUTPUT is assumed.
n	Number of files (decimal) to copy; if this parameter is omitted, n=1 is assumed.

The COPYSBF routine copies n files beginning at the current position of lfn_1 to file lfn_2, shifting each line image one character to the right and adding a leading space. Each line image may contain up to 150 (6-bit) characters. Any characters beyond 150 are lost. A page eject character is inserted at the beginning of each logical record (refer to appendix I for a list of carriage control characters). If lfn_1=lfn_2, n files are skipped but no data transfer occurs. If the EOI is encountered before the file count is satisfied, an EOF is written to lfn_2, and the operation terminates.

If COPYSBF is attempted on a line longer than 150 (6-bit) characters, the following message is issued.

NO LINE TERMINATOR.

COPYX STATEMENT

The COPYX control statement enables the user to specify certain conditions when copying logical records.

NOTE

The COPYX statement produces unpredictable results when copying S, L, or F format tapes. The COPY utility is recommended for copying tapes in these formats.

The control statement format is:

COPYX(lfn_1, lfn_2, x, b, c)

lfn_1 Name of the file to copy from; if this parameter is omitted, file INPUT is assumed.

lfn_2 Name of the file to copy to; if this parameter is omitted, file OUTPUT is assumed.

x Copy specifications; if omitted, one record is copied. The value for x may be one of the following:

x	Meaning
n	Number of records (decimal) to copy.
00	Copy all records up to and including first zero-length record.
name	Copy all records up to and including record of specified name (record name is first 7 characters of record or the name in the prefix table, if present).
type/name	Copy all records up to and including record of specified type and name (refer to Library Record Types in section 14 for list of valid record types).

b Backspace control; if omitted, 0 is assumed.

b	Meaning
0	No backspace.
1	Backspace file lfn_1 one record after copy completes.
2	Backspace file lfn_2 one record after copy completes.
3	Backspace files lfn_1 and lfn_2 one record after copy completes.

c If a fifth parameter (1 to 7 alphanumeric characters) is present, the copy to or from an S or L format tape is performed in coded rather than binary mode. If coded mode is set on an SI tape, the system aborts the job. For other formats, the mode setting is ignored.

The COPYX routine copies logical records from lfn_1 to file lfn_2 at the current position of lfn_1 until the condition specified by x is met. It then backspaces the files according to the value specified by the b parameter. If an EOF or EOI is encountered on lfn_1 before the condition specified by x is met, the operation terminates and the backspace parameter b is ignored. If $lfn_1=lfn_2$, the file is read until the termination condition is satisfied or an EOF or EOI is encountered.

If EOI is encountered on lfn_1 before the termination condition is satisfied, an additional EOR is written on lfn_2 only if data has been transferred since the previous EOR was written (or since the beginning of the copy if no EORs have been encountered).

DISPOSE STATEMENT †

The DISPOSE control statement releases the specified files to the named output queues.

The control statement format is:

DISPOSE($lfn_1=q_1, lfn_2=q_2, \ldots, lfn_n=q_n$/ot=usernum)

lfn_i Name of the file to be disposed. lfn cannot be a direct access file or the primary file.

q_i Queue type:

PR	Print
PH	Punch coded O26
P9	Punch coded O29
PB	Punch binary
P8	Punch 80-column binary

ot Origin type to which files are to be disposed:

BC	Local batch
EI	Remote batch

usernum Number of the remote batch (that is, ot is EI) user to which the files are to be disposed (ignored if ot is BC). This parameter is valid only if the user is allowed deferred batch jobs. Also, usernum must match the number of the user performing the DISPOSE on all character positions except those containing an *.

The file type for file lfn_i is changed to q_i in the FNT/FST entry for lfn_i. The system then processes the file according to queue type. The user can dispose coded punch files to either O26 or O29 regardless of the job's initial keypunch mode. If the system cannot recognize q_i, the following message is issued.

ILLEGAL DISPOSE CODE.

If the ot and usernum parameters are not specified, a remote batch job disposes the files to the remote terminal from which it was submitted, and all other origin types dispose the files to the central site output device. If ot is BC, the usernum parameter is ignored, and the files are disposed to the central site device.

---

† The user should employ the ROUTE control statement for this operation.

DOCMENT STATEMENT

The DOCMENT control statement enables the user to extract either the external or internal documentation from a file containing COMPASS source code.

The control statement format is:

DOCMENT($p_1, p_2, \ldots, p_n$)

p_i The parameters can be in any order and must be in one of the following forms.

Omitted	The first default value is assumed.
a	The alternate default value is assumed.
a=x	x is substituted for the assumed value.

Any numeric parameter can be specified with a postradix character of either B or D. The values that p_i can assume are:

I=lfn_1 Name of the file that contains the page footing information; this must be a single statement in the following format.

Column(s)	Contents
1	Blank
2-45	Document title
46-55	Publication number
56-60	Revision level
61-70	Revision date

S=lfn_2 Name of the file containing the source statement images from which to extract the documentation. This file is rewound by default unless the NR parameter is specified.

L=lfn_3 Name of the file on which the output is to be written.

N=nn Number of copies to be produced.

T=type Documentation type:

INT	Internal documentation (detailed description of the internal features of the software).
EXT	External documentation (detailed description of the external features of the software).

C=cc Key character for documentation.

P=pp Number of print lines per page.

NR Disable rewind on the S (source) file.

NT Negate table generator.

TC List table of contents.

The following are the default values for the parameters described.

Parameter	First Default	Alternate Default	Comment
I	0	INPUT	Page footing information; if I is 0, no footing information is printed.
S	COMPILE	SOURCE	Source statement images.
L	OUTPUT	OUTPUT	List file.
N	1	1	Number of copies (decimal).
T	EXT	INT	Documentation type.
C	*	03	Check character (two octal digits).
P	60	80	Number of print lines per page.
NR	REWIND	NO REWIND	Source file rewind status.
NT	ON	OFF	Table generator status.
TC	OFF	ON	Table of contents status.

Refer to appendix I in volume 2 for a detailed explanation of the documentation standards followed. This appendix also contains an example of external and internal documentation for a sample program.

EVICT STATEMENT

The EVICT control statement releases file space for the specified files but does not release file assignment to the job.

The control statement format is:

EVICT($lfn_1, lfn_2, \ldots, lfn_n$)

lfn_i Name(s) of the file(s) to be evicted.

The operation that EVICT performs depends on the file characteristics.

File	EVICT Action
Permanent file	Releases all file space except the first track and writes an EOI on the first sector of the first track, but keeps file assigned to job.
Deferred routed queue file†	Releases all file space and clears all file routing information.
File with write interlock set	Unloads file.
All other files	Writes file length on first sector of first track and releases file space, but keeps file assigned to job.

† Refer to the ROUTE statement in this section.

An EVICT of a tape file performs the same function as an UNLOAD and so cannot be used to decrease the number of resource units scheduled via the RESOURC statement.

FCOPY STATEMENT

The FCOPY control statement converts a file from one code set to another. Currently, the only supported conversion is from 6/12 display code (used in time-sharing ASCII mode) to 12-bit ASCII code. Refer to appendix A for code set definitions.

The control statement format is:

FCOPY(P=lfn_1,N=lfn_2,PC=cs_1,NC=cs_2,R)

P=lfn_1	File to be converted (default is OLD). The user should assign lfn_1 to the job before performing the FCOPY operation.
N=lfn_2	File on which the converted data from lfn_1 is written (default is NEW). If lfn_2 is not assigned to the job, FCOPY creates it.
PC=cs_1	Code set of lfn_1. The default and only current supported value for cs_1 is ASCII, which refers to 6/12 display code.
NC=cs_2	Code set of lfn_2. The default and only currently supported value for cs_2 is ASCII8, which refers to 12-bit ASCII code.
R	If R is specified, lfn_1 and lfn_2 are rewound before and after the conversion. If R is omitted, lfn_1 and lfn_2 are not rewound before or after the conversion.

FCOPY reads lfn_1 to its EOI, preserving its EOR and EOF marks on the converted file. The maximum line length that can be processed is 160 12-bit codes or 320 6-bit codes. Lines that exceed the maximum length are truncated.

NOTE

If lfn_1 is written in 6/12 display code based on the 63-character set, it must be converted to the 64-character set by the CONVERT control statement before its conversion by the FCOPY statement.

Files converted to 12-bit ASCII code can be listed on a local batch printer (refer to the ROUTE control statement) but cannot be listed at a time-sharing or remote batch terminal.

Example:

A time-sharing user wants to print a file (FILE1) created in ASCII mode. To do so, he enters a COPYSBF statement to prefix the file lines with appropriate carriage control characters. He then enters an FCOPY statement to convert the file containing 6/12 display code (FILE2) to a file containing 12-bit ASCII code (FILE3). Finally, he routes the converted file (FILE3) to a line printer that prints the ASCII graphic 95-character set.

```
/ascii
/copy,file1.
AaBbCcDdEeFfGg
HhIiJjKkLlMmNn
 EOI ENCOUNTERED.
/rewind,file1.
 $REWIND,FILE1.
/copysbf,file1,file2.
 END OF INFORMATION ENCOUNTERED.
/rewind,file2.
 $REWIND,FILE2.
/copy,file2.
1AaBbCcDdEeFfGg
 HhIiJjKkLlMmNn
 EOI ENCOUNTERED.
/rewind,file2.
 $REWIND,FILE2.
/fcopy,p=file2,n=file3,r.
 FCOPY COMPLETE.
/route,file3,dc=lp,ec=a9.
 ROUTE COMPLETE.
```

The following is the line printer output from the ROUTE statement.

```
AaBbCcDdEeFfGg
HhIiJjKkLlMmNn
```

LIST80 STATEMENT

The LIST80 routine reads a file containing list output produced by the COMPASS assembler and compresses it to 80 columns, which fits on 8-1/2-by 11- inch printer paper.

The control statement format is:

LIST80(lfn_1, lfn_2, NR)

lfn_1	File to copy from; if this parameter is omitted, file LIST is assumed.
lfn_2	File to copy to; if this parameter is omitted, file OUTPUT is assumed.
NR	If this parameter is specified, lfn_1 is not rewound.

LOCK STATEMENT

The LOCK control statement enables the user to prevent writing on a file.

The control statement format is:

LOCK(lfn_1, lfn_2, . . . , lfn_n)

lfn_i	Logical file name of a local file.

With the LOCK statement, the user can set the write interlock bit in the FNT/FST entry for a local file. Subsequently, the system allows only read operations on the file. The file specified must be a local file; if it is not, the following message is issued.

ILLEGAL FILE TYPE.

The LOCK statement may also be used in conjunction with the COMMON statement to lock local files before making them library files for multiple user access. Refer to Library Files in section 2 and the COMMON control statement in this section.

LO72 STATEMENT

The LO72 control statement allows the user to specify the reformatting of his files.

The control statement format is:

LO72($p_1, p_2, \ldots, p_n$)

p_i Any of the following parameters in any order:

I — Reformat parameters are on file INPUT.
I=lfn_1 — Reformat parameters are on file lfn_1.
I=0 — There is no input file of reformat parameters.
If the I parameter is omitted, I=0 is assumed.

S — Data to be reformatted is on file SCR.
S=lfn_2 — Data to be reformatted is on file lfn_2. If the S parameter is omitted, SCR is assumed.

L — Reformatted data is listed on file OUTPUT.
L=lfn_3 — Reformatted data is listed on file lfn_3. If the L parameter is omitted, OUTPUT is assumed.

T — File to be reformatted is of type B.
T=x — File to be reformatted is of type x, where x is:

- M Modify source data
- C COMPASS source data
- B Other source data

If the T parameter is omitted, B is assumed.

H — Number of characters per output line is 72.
H=xxx — Number of characters per output line is xxx (maximum allowed is 150 characters). If the H parameter is omitted, 72 is assumed.

NOTE

H must be greater than or equal to the number of characters being moved (Nx) plus the starting column number of the destination field (Ox).

LP — Output is formatted for the line printer.

NR — Output file is not rewound.

Nx=y — Specifies the number of characters to be moved (up to 6 fields):

- x(1 to 6) — Number of the field being moved
- y — Number of characters being moved

NOTE

The follwing restrictions apply to the H, N, I,and O parameters:

(Nx+Ix). GT. 150	Yields an error
(Nx+Ox). GT. H	Yields an error
H. GT. 150	Yields an error
	where $1 \leq x \leq 6$

Lx=y — Specifies the field the data originates from:

- x(1 to 6) — Number of the field being moved
- y — Starting column of originating field

Ox=y — Specifies the destination field the data is going to:

- x(1 to 6) — Number of the field to receive data
- y — Starting column of destination field

IT — Suppresses query to terminal asking if user wishes to change any of the input parameters before processing begins. If omitted, query is issued. This parameter is effective only from time-sharing origin jobs.

The following shows the default values assumed for the N, O, and I parameters for the various source types.

Type	N1	I1	O1	N2	I2	O2	N3	I3	O3
B	72	1	1	0	0	0	0	0	0
C	7	9	1	50	41	8	15	112	58
M	2	6	1	48	10	3	22	82	51

The remaining parameters of these types are defaulted to 0.

LO72 reformats files (output files in general). The user can rearrange each line (all lines must be formatted the same) in the format he chooses. All default values compress output to 72 columns, which is appropriate for terminal output or 8-1/2 by 11-inch printer paper. If a 1 is encountered in column 1 (the page eject printer control character), the next two lines of source data are processed as a two-line header. This header is compressed to 72 columns for all source types. If no page eject control characters are encountered, no headers are processed.

The following values apply to the first line of header and cannot be changed.

N1=42, I1=8, O1=0 (if LP not specified; otherwise, O1=1)
N2=20, I2=90, O2=42
N3=5, I3=115, O3=62
N4=5, I4=121, O4=67

The subheader lines for COMPASS and Modify listings are processed uniquely.

For B listings, the following values apply to the reformatting.

N1=43, I1=8, O1=0 (if LP not specified; otherwise, O1=1)
N2=29, I2=70, O2=43

All parameters are passed to LO72 by the control statement. If an input file is specified, LO72 reads it for additional input parameters. If the job originates from a time-sharing terminal, and the IT parameter is not specified, the user is asked if he wishes to change any of the input parameters. If he enters YES, the system prints the current parameter values and allows him to change them individually. Pressing the carriage return key for any parameter leaves the parameter at its former value. In the following examples, the same input parameters are entered in three possible ways.

Control Statement:

```
LO72(I=0,S=SOURCE,T=B,L=OUT,N4=1,I4=2,O4=75,H=90)
```

Time-Sharing Terminal: (User entries are in lowercase. The symbol CR indicates carriage return.)

```
/lo72
DO YOU WANT TO CHANGE ANY CONTROL ARGUMENT VALUES-
ENTER: YES OR NO
? yes CR
ARGUMENT              VALUE
INPUT FILE NAME:              ? CR
SOURCE FILE NAME:     SCR     ? source CR
OUTPUT FILE NAME:     OUTPUT  ? out CR
SOURCE FILE TYPE:     BATCH ? b CR
OUTPUT LINE LENGTH: 72  CHARS.? 90 CR
   NO. OF  MOVED FROM  MOVED TO
   CHARS.    COLUMN     COLUMN
(X) (NX)      (IX)       (OX)
 1.  72        1          1
 2.  0         0          0
 3.  0         0          0
 4.  0         0          0
 5.  0         0          0
 6.  0         0          0
ENTER CHANGES IN THE FOLLOWING FORMAT:
NX=AA*CR*
IX=BB*CR*
OX=CC*CR*
ETC.
TO CONTINUE, ENTER *CR* ONLY. ? n4=1 CR
? i4=2 CR
? o4=75 CR
? CR
 LO72 COMPLETE.
```

Input File: (Each line in the input file must end with a terminator.)

```
S=SOURCE, L=OUT, T=B.
N4=1, I4=2, O4=75.
H=90.
-EOR-
```

NEW STATEMENT

The NEW control statement creates a primary file.

The control statement format is:

NEW(lfn/ND)

lfn	Name of file to be made primary file.
ND	If this parameter is specified, other files currently assigned to the job are not released.

The NEW statement creates an empty file and makes it the user's new primary file. Any currently existing primary file is released.

All files assigned to the job are released unless the ND parameter is specified.

Refer to the note in PRIMARY Statement in this section for use of primary file types.

OUT STATEMENT

The OUT control statement releases output files from the control point to the output queue.

The control statement format is:

OUT.

The only files released are those having the names

OUTPUT
PUNCH
PUNCHB
P8

or any local files belonging to one of these types. An example would be any of the above files that had been renamed.

The number of files released is recorded in the job's dayfile with the message

xx FILE(S) RELEASED.

where xx is the octal number of files released.

If no files with the above names or belonging to these types are found, the following message is issued to the dayfile:

NO FILE(S) RELEASED.

This control statement is used if the user wishes to initiate printing or punching of the files before job termination. The PUNCH file is punched in either O26 or O29 mode, depending on the origin of the job. If the job is a local batch job, the coded deck is punched in the initial keypunch mode of the job's control statement record. For all other job origin types, the coded file is punched in the system default keypunch mode.

PACK STATEMENT

The PACK control statement removes all EOR and EOF marks from a specified file and copies it as one record to another file.

The control statement format is:

PACK(lfn_1, lfn_2, x)

lfn_1	Name of file to be packed.
lfn_2	Name of file to receive packed data.
x	If a third parameter (1 to 7 alphanumeric characters) is specified, lfn_1 is not rewound before the pack occurs.

The input file, lfn_1, may consist of any number of records and/or files. If no third parameter is supplied, lfn_1 is read from the BOI to the EOI, and all EOR and EOF marks are removed. It is written to file lfn_2 at the current position as one record. File lfn_2 is rewound after the pack; lfn_1 is not. If lfn_2 is not specified, file lfn_1 is packed to itself.

PRIMARY STATEMENT

The PRIMARY control statement makes a local file the primary file, or it creates an empty primary file.

The control statement format is:

PRIMARY(lfn)

lfn	Name of local file.

If lfn already exists, it must be a local mass storage file in order to be made the primary file. If lfn does not exist, the PRIMARY statement creates it on mass storage. Any currently existing primary file (other than the lfn specified) is released. If the specified file is already primary, the operation is ignored.

NOTE

The primary file is rewound before every operation performed on that file. Therefore, the file manipulation statements BKSP, SKIPEI, SKIPF, SKIPFB, and SKIPR cannot be used to position within the file. The user should also remember that the primary file is rewound after the completion of any of the COPY statements. An attempt to add to the file using one of the COPY statements may result in writing over existing data at the BOI.

RENAME STATEMENT

The RENAME control statement allows the user to change the name of a local file.

The control statement format is:

RENAME($nlfn_1$=$olfn_1$, $nlfn_2$=$olfn_2$, . . . , $nlfn_n$=$olfn_n$)

$nlfn_i$ New name of the local file.

$olfn_i$ Existing name of the local file.

The RENAME control statement changes the name of the file $olfn_i$ to $nlfn_i$ in the FNT/FST. This does not change the names of files in the permanent file system. Normally, the file type of nlfn is the same as the file type of olfn.

If a file by the name $nlfn_i$ already exists, it is returned to the system. Under certain conditions, the system also changes the file type of $olfn_i$ to that of the file which was returned.

- If $olfn_i$ is a local mass storage file and the returned file was a print, punch, or primary type file, $olfn_i$ is renamed and its file type is changed to that of the returned file.
- If $olfn_i$ is a local mass storage file and the returned file was not a print, punch, or a primary type file, $olfn_i$ is renamed but its file type is not changed.
- If $olfn_i$ is not a local file and nlfn and olfn are not the same file types or if $olfn_i$ does not reside on mass storage, an

  ILLEGAL FILE TYPE.

  error message is issued.

For example, the user has only two files assigned to his job. File A is a local mass storage file, and file B is a print type file. If the user issues the following request

RENAME(X=A)

file A is renamed file X, and its file type (local) is not changed. However, if the user issues the request

RENAME(B=A)

file B is returned to the system; file A is renamed file B and changed to print type file.

REQUEST STATEMENT

The REQUEST statement sends a message to the system operator requesting that the named file be assigned to the device described in the comment field.

The control statement format is:

REQUEST(lfn, {CK / CB})comment

lfn — Name of the file to be assigned to the specified equipment.

CK — Specifies that lfn is to be used as a checkpoint file. Each time a checkpoint dump is taken, the new information is written at the previous EOI of lfn.

CB — Specifies that lfn is to be used as a checkpoint file. Each time a checkpoint dump is taken, the new information is written at the BOI of lfn.

If lfn already exists when the REQUEST is made, no new assignment is made and job processing continues with the next control statement. However, the user can reassign lfn by issuing a RETURN on the file before making the REQUEST.

Any user, regardless of his validation, may use the REQUEST statement to assign a file to a mass storage device. However, to assign a file to a nonmass storage device, the user must be validated to use nonallocatable devices.† If the user does not have this validation and attempts to request a nonmass storage device, the system aborts his job.

If lfn is to be used for checkpoint dumps, either the CK or CB keyword is specified. These keywords are used in conjunction with the CKP and RESTART control statements; they allow the user to:

- Save all checkpoint dumps by appending each dump to the checkpoint file:

  REQUEST(lfn, CK)

- Save the last checkpoint dump by writing each dump at the beginning of the checkpoint file:

  REQUEST(lfn, CB)

- Save two consecutive checkpoint dumps by alternately writing on two checkpoint files:

  REQUEST(lfn_1, CB)
  REQUEST(lfn_2, CB)

If the CK parameter is specified for alternate files or if more than two checkpoint files are specified, the job is aborted and the following message is issued to the user's dayfile.

CHECKPOINT FILE ERROR.

† Refer to LIMITS control statement in section 6.

The CK and CB parameters specify a checkpoint file that is local to the job. The user can make the checkpoint file permanent by placing a DEFINE statement† before the REQUEST.

```
DEFINE(lfn)
REQUEST(lfn, CK)
CKP.
```

The user is not required to supply a REQUEST statement to define a checkpoint file. He can use an ASSIGN or LABEL statement or he can use default values.

If no REQUEST statement specifying a checkpoint file has been detected when the first CKP statement is encountered, the system requests a device for the user, specifies a file name of CCCCCCC, and selects the CK option. For a subsequent restart job, however, the system assumes the user has made the checkpoint file available.

The REQUEST statement can also be used to create or access existing 7- or 9-track unlabeled tapes. If a magnetic tape assignment is needed to satisfy a REQUEST, the MT or NT parameter should be specified. For a description of magnetic tape assignment with the REQUEST statement, refer to Tape Management in section 10.

RESEQ STATEMENT

The RESEQ control statement is used to resequence source files which have leading sequence numbers or to add sequence numbers to an unsequenced file.

The control statement format is:

```
RESEQ(lfn, t, xxx, yy)
```

lfn	Name of the sorted file to be resequenced. RESEQ does not sort lfn (refer to the SORT statement).	
t	Type of file:	
	B	BASIC source code.
	T	Text source information; a five-digit sequence number plus a blank is added at the beginning of each line; the file text, however, is not inspected.
	other or omitted	Any number at the beginning of a line is considered a sequence number and is resequenced according to the xxx and yy parameters, numbers are added to lines where no leading sequence numbers are present. This option can be used with time-sharing FORTRAN statements.
xxx	New line number of the first statement; if this parameter is omitted, the system assumes xxx=100.	
yy	Increment to be added to xxx for each succeeding line number; if this parameter is omitted, the system assumes yy=10.	

† Any mass storage file used as a checkpoint file must have write permission.

Files which have leading sequence numbers include time-sharing FORTRAN and BASIC source files. If the file has no leading sequence numbers, five-digit numbers are inserted at the beginning of each line. If the line number encountered or required exceeds 99999, RESEQ issues an error message.

When resequencing a BASIC source program, the user must specify B for the file type parameter, t, so that RESEQ changes the line number references within the source statements. RESEQ supplies five-digit line numbers and line number references; excess surrounding blanks are used in the expansion of line number references.

Example:

File X contains the following BASIC source statements.

```
95 ON SGN(A)+2  GOTO 100,110,120      'COMMENT
100 PRINT "A IS NEGATIVE"
105     GOTO    130                   'COMMENT
110 PRINT "A IS ZERO"
115     GOTO    130                   'COMMENT
120 PRINT "A IS POSITIVE"
130 LET B=A+1
135 END
```

The following statement changes the contents of file X.

```
RESEQ(X,B,90,10)
```

The user then rewinds and lists file X.

```
00090 ON SGN(A)+2  GOTO 00100,00120,00140 'COMMENT
00100 PRINT "A IS NEGATIVE"
00110     GOTO  00150                   'COMMENT
00120 PRINT "A IS ZERO"
00130     GOTO  00150                   'COMMENT
00140 PRINT "A IS POSITIVE"
00150 LET B=A+1
00160 END
```

The RESEQ statement changes the line numbers and the line number references. Line numbers now begin at 90 and increment by 10. The comment on the first line is moved to the right to allow for the expanded line number references.

RETURN STATEMENT

The RETURN control statement releases files assigned to a job and may release file space depending on the file type.

The control statement formats are:

RETURN($lfn_1, lfn_2, \ldots, lfn_n$)

or

RETURN(*, $lfn_1, lfn_2, \ldots, lfn_n$)

The first format returns the named files ($lfn_1, lfn_2, \ldots, lfn_n$). The second format returns all files assigned to the job except the named files. If no files are named on the second format the asterisk specification returns all files assigned to the job. An error message is returned if neither an asterisk nor a file name is specified.

RETURN performs the following operations according to the file type.

Type	Operation
Input	The file name is changed to INPUT*. File space is not released (refer to Input File Control in section 3 for further information).
Print	File space is released, and the file is no longer assigned to the job. (The file is not printed.)
Punch	File space is released, and the file is no longer assigned to the job. (The file is not punched.)
Local	File space is released, and the file is no longer assigned to the job.
Primary	Same as Local.
System	File space remains, but the file is no longer assigned to the job.
Library	File space remains, but the file is no longer assigned to the job.
Direct access	The write interlock is cleared. File space remains, but the file is no longer attached to the job.

In addition, the RETURN of a magnetic tape file or the RETURN of the user's last direct access file on an auxiliary removable disk pack decrements the resource demand count as scheduled by the RESOURC control statement if, and only if, the total concurrent resource demand (tapes and removable packs) has been satisfied.

To release a file without decrementing the resource demand count, the user can issue an UNLOAD statement. To release file space without releasing the file from the job, the user can issue an EVICT statement.

REWIND STATEMENT

The REWIND control statement rewinds files. A mass storage file is positioned at its BOI. An unlabeled tape file is positioned at its load point. A labeled tape file is positioned after the first HDR1 label for the file. If the tape file begins on a previous volume, the system notifies the operator to mount that volume.

The control statement formats are:

REWIND($lfn_1, lfn_2, \ldots, lfn_n$)

or

REWIND(*, $lfn_1, lfn_2, \ldots, lfn_n$)

The first format rewinds the named files ($lfn_1, lfn_2, \ldots, lfn_n$). The second format rewinds all files assigned to the job except the named files. If no files are named on the second format, the asterisk specification rewinds all files assigned to the job.

If the previous operation on the magnetic tape file was a write, a REWIND statement causes the following operations to be performed.

1. If the tape is ANSI labeled, the system writes a tape mark, an EOF1 label, and three tape marks and then rewinds the tape.
2. If the tape is unlabeled and the data format specified on the ASSIGN, LABEL, or REQUEST statement is S, L, or F, the system writes four tape marks and then rewinds the tape.
3. If the tape is unlabeled and the data format is I or SI, the system writes a tape mark, an EOF1 label, and three tape marks and then rewinds the tape.

Refer to Magnetic Tape Files in section 2 and to Tape Management in section 10 for further information about tape files and to appendix G for a description of EOF1 and EOV1 labels.

ROUTE STATEMENT

The ROUTE control statement prepares a designated file for release to an input or output queue. The file routing requested may take effect when the statement is processed, or it may be deferred. If deferred, the routing characteristics specified define the handling of the file in later job steps or at job termination. This statement also allows the user to rescind a prior deferred ROUTE statement, changing the file type to local.

The control statement format is:

ROUTE(lfn, $p_1, p_2, \ldots, p_n$)

Descriptions of the statement parameters follow. The lfn parameter is required on all ROUTE statements.

lfn	Name of the file to route. lfn can be an input, print, punch, or local file; it cannot be a primary or direct access file.

The remaining parameters are order-independent.

p_i	Description
DC=xx	Disposition code; assumes any one of the following 2-character codes.

IN Release file to input queue. Normal job input file format is required. If the job statement within the file is in error, the file is not released and remains a local file. ROUTE issues a dayfile message explaining the error.

Print codes:

LP	Print on any printer
PR	Same as LP
LR	Print on 580-12 printer
LS	Print on 580-16 printer
LT	Print on 580-20 printer

Punch codes:

SB	Punch system binary
PB	Same as SB
P8	Punch 80-column binary
PU	Punch coded
PH	Same as PU

PL Plotter

SC Rescind prior routing and change the file type to local

If the DC parameter is omitted and lfn is a deferred routed file (refer to the FM parameter), the disposition code previously specified remains in effect. If the DC parameter is omitted and lfn is not a deferred routed file, the default depends on the file name specified for lfn.

If DC is omitted and lfn is:	ROUTE assumes DC is:
OUTPUT	DC=LP
PUNCH	DC=PU
PUNCHB	DC=SB
P8	DC=P8
Any other name	DC=SC

p_i	Description
DEF	Indicates that routing of the file to the queue is deferred to a later job step or end of job. If this parameter is specified, the file is created if it does not exist. DEF is not allowed if DC=IN.
EC=xx	Defines external characteristics for print or punch files.

For print files, xx can be the following.

A4	Provided for NOS/BE compatibility.
A6	ASCII 64-character set.
A9	ASCII 95-character set.
B4	Provided for NOS/BE compatibility.
B6	Display code 63/64-character set.

For punch files, xx can be the following.

ASCII	Punch ASCII.
O26	Punch O26 mode.
O29	Punch O29 mode.
SB	Punch system binary.
80COL	Punch 80-column binary.

NOTE

If an invalid external characteristic is specified, the queue file processor cannot output the file. The user must not specify a print file characteristic for a punch file or a punch file characteristic for a print file. He also must not specify an external characteristic not available at the site. If EC is not specified, an appropriate EC default is set on the basis of the DC parameter setting and installation options.

FC=xx	Forms code; specifies routing to the output device that the system operator assigned the forms code xx. This parameter prevents output of a file before its special forms are placed in the output device. xx can be any 2 alphanumeric characters, but the combinations null, AA, AB, AC, AD, AD, AE, and AF give maximum system efficiency. A value of null results when no FC parameter is specified.
FID=xx	An NOS/BE parameter included for compatibility. It produces an informative message under NOS.
FM	Implicit remote routing (refer to the following note).

p_i	Description
FM = xx	1 to 7 alphanumeric character family name; indicates routing to a remote batch terminal logged in with the specified family name. The note following the ROUTE parameter descriptions describes the default procedures.
IC=xx	Internal characteristics; specifies one of the following. DIS Display code ASCII ASCII code BIN Binary This parameter is normally not specified since its default is automatically established through the disposition code DC.
ID=xx	Selects local device ID from 0 to 67 (octal default). This is identical to the ID specified by the SETID control statement.
ID	Implicit central site routing (refer to the note at the end of the parameter descriptions).
PRI=xx	File priority. This is a NOS/BE parameter included for compatibility. It produces an informative message under NOS.
REP=xx	The number of additional file copies to be routed to a destination. The range for xx is from 0 to 31; therefore, the number of copies that can be sent ranges from 1 to 32. Values for xx beyond its range are set to zero, an informative message is set, and one copy is routed to the destination.
SC=xx	Spacing code for the 580-PFC printer. This is a numeric value from 0 to 77 (octal default).
ST=xx	Station ID. This is a NOS/BE parameter included for compatibility. It produces an informative message under NOS.
TID	Implicit remote routing (refer to the note at the end of the parameter descriptions).
TID=C	Central site routing, This is a NOS/BE parameter included for compatibility. Its action is identical to the ID parameter.
TID=xx	Terminal ID. This form of the TID parameter is included for NOS/BE compatibility. Under NOS, it is processed the same as TID; however, an informative message is issued stating that xx is ignored.
UN	Implicit remote routing (refer to the note at the end of the parameter descriptions).
UN=xx	Specifies the user number of the remote batch user to whom the named file is routed. The parameter xx is valid only if it matches the user number of the user performing the route. The matching is character for character except for those positions containing an * (refer to the note at the end of the parameter descriptions).

NOTE

For remote batch origin (EIOT) jobs, the following action is taken.

- Parameter ID, ID=xx, or TID=C causes routing to the central site.
- Parameter FM, TID, or UN with no argument causes routing to the terminal of origin.
- The omission of FM, TID, or UN causes routing to the terminal of origin.
- Parameter FM or UN with legal arguments causes routing to the specified terminal.

For jobs of any origin other than EIOT, the following action is taken.

- Parameters ID, ID=xx, and TID=C causes routing to the central site.
- Specifying UN, TID, or FM without parameters causes routing to the terminal specified by the job's FM and UN at the time of the ROUTE call.
- Specifying UN or FM with legal arguments causes routing to the selected remote terminal.

If a job is routed to the input queue with an illegal USER control statement, the following message is issued

```
DSP - ILLEGAL USER CARD.
SYSTEM ABORT.
```

and the job is aborted with no error exit processing or if submitted from a terminal, the terminal is logged off. The security count for the user number that did the ROUTE is decremented accordingly.

SETID STATEMENT†

The SETID control statement assigns a new identification code for the specified file.

The control statement format is:

SETID(lfn_1=x_1,lfn_2=x_2,...,lfn_n=x_n)

lfn_i	Logical file name.
x_i	New identification code for the file (0 through 67_8). This code must match the device identification code specified in the EST. (The installation establishes the device identification codes.)

The identification code allows the user to route his file to an output device or device group with the same identification code. This is useful when a print file requires special forms.

The file LFN_i must be an input (INFT), local (LOFT), print (PRFT), or punch (PHFT) type file, or the following message is issued.

```
ILLEGAL FILE TYPE
```

† The ROUTE control statement should be used to perform this operation.

SKIPEI STATEMENT

The SKIPEI control statement directs the system to position the specified file at the EOI.

The control statement format is:

SKIPEI(lfn)

lfn	Name of the file to be positioned.

On magnetic tapes where no EOI is defined, the operation stops at an EOF.

The SKIPEI statement has no effect on a primary file since the file is rewound before every operation.

SKIPF STATEMENT

The SKIPF control statement directs the system to bypass, in a forward direction, the specified number of files from the current position of the named file.

The control statement format is:

SKIPF(lfn, n, m)

lfn	Name of the file to be positioned.
n	Number (decimal) of files to be skipped; if the parameter is omitted, the system assumes n=1.
m	File mode: C for coded, B for binary. If omitted, the system assumes the file is in binary mode. If coded mode is set on an SI tape, the system aborts the job.

If an EOI is encountered before n files are bypassed, file lfn remains positioned at the EOI.

The SKIPF statement has no effect on a primary file since the file is rewound before every operation.

SKIPFB STATEMENT

The SKIPFB control statement directs the system to bypass, in the reverse direction, the specified number of files from the current position of the named file.

The control statement format is:

SKIPFB(lfn, n, m)

lfn	Name of the file to be positioned.
n	Number (decimal) of files to be skipped; if the parameter is omitted, the system assumes n=1.
m	File mode: C for coded, B for binary. If omitted, the system assumes the file is in binary mode. If coded mode is set on an SI tape, the system aborts the job.

The system does not backspace past the beginning-of-information (BOI) or load point (tape file) in the event that BOI or load point is encountered before n files are bypassed.

The SKIPFB statement has no effect on a primary file since the file is rewound before every operation.

SKIPR STATEMENT

The SKIPR control statement directs the system to bypass, in a forward direction, the specified number of logical records from the current position of the named file.

The control statement format is:

SKIPR(lfn, n, ℓ, m)

lfn	Name of the file to be positioned.
n	Number (decimal) of records to be skipped; if this parameter is omitted, the system assumes n=1.
ℓ	EOR level; $0 \leq \ell \leq 17$. If $0 \leq \ell \leq 16$, the system assumes $\ell = 0$. If $\ell = 17$, n indicates the number of files to skip rather than records.
m	File mode: C for coded, B for binary. If omitted, the system assumes the file is in binary mode. If coded mode is set on an SI tape, the system aborts the job.

EOR marks are considered separate records and included in the record count. If the EOI is encountered before n records are bypassed, file lfn remains positioned at the EOI.

The SKIPR statement has no effect on a primary file since the file is rewound before every operation.

SORT STATEMENT

The SORT control statement enables the user to sort a file of line images or statements in numerical order based on leading line numbers consisting of a specified number of digits.

The control statement format is:

SORT(lfn, NC=n)

lfn	Logical file name of the file to be sorted; lfn may be a local file or a direct access permanent file.
n	Number of leading line number digits on which the file is to be sorted; $n \leq 10$. If the NC parameter is omitted, the system assumes n=5.

In the case of duplicate line numbers, all lines other than the first are considered correction lines. All lines with the same number are deleted from the file except the last line encountered.

For input from a time-sharing terminal, SORT deletes a line image or statement if a line number is followed by an empty line or a line number is followed by a blank and a carriage return.

For batch input, SORT deletes a statement or line image if a card containing only the line number is submitted.

If a line number contains more than n digits, the user can delete the line either by entering the first n digits of the line number and pressing the carriage return (terminal input) or by submitting a card containing only the first n digits of the line number (batch input).

After the sort, lfn is packed and set at EOI.

TCOPY STATEMENT

The TCOPY control statement copies X (external) format binary tapes or E (line image), B (blocked), or SI (system internal) format coded tapes to mass storage, to an I format tape, or to an SI binary format tape. It also writes E or B format tapes converted from files on mass storage, I format tape, or SI format binary tape. The X binary and E, B, and SI coded tape formats were supported under earlier versions of NOS. Now, to access data or write data in one of these formats, the tape must be assigned as an S (stranger) format tape (refer to the tape assignment statements in section 10) and the file copied using the TCOPY statement.

The parameters on the TCOPY control statement can appear in order-dependent format, order-independent format, or a combination of both. The completely order-dependent format is:

TCOPY(lfn_1, lfn_2, format, tc, copycnt, charcnt, erlimit, p_1p_2, lfn_3)

The completely order-independent format is:

TCOPY(I=lfn_1, O=lfn_2, F=format, TC=tc, N=copycnt, CC=charcnt, EL=erlimit, PO=p_1p_2, L=lfn_3)

If order-dependent and order-independent parameters are mixed in one TCOPY statement, the order-dependent parameters must appear in their proper position. All parameters are optional. However, the specification of certain parameters precludes the application of others. A nonapplicable parameter may be ignored or it may be illegal. This is stated in the individual descriptions of the parameters.

The parameters are defined as follows:

Parameter	Description	Default
I=lfn_1	Name of the file to copy from.	INPUT
O=lfn_2	Name of the file to copy to.	OUTPUT
F=format	Data format that specifies the type of conversion for the copy operation. This can be any one of the following.	X

Parameter	Description		Default
	format	Conversion	
	E	Copy an E format tape to mass storage, an I, or an SI binary tape file, or generate a new E format tape from mass storage, an I, or an SI binary tape file. The E tape must be unlabeled and assigned as S format.	
	B	Copy a B format tape to mass storage, an I, or an SI binary tape file, or generate a new B tape from mass storage, an I, or an SI binary tape file. The B tape must be unlabeled and assigned as S format.	
	X	Copy an X format tape to mass storage, an I, or an SI binary tape file. The unlabeled input tape must be assigned an S format, with noise size of 8 for 7-track or 6 for 9-track tape (refer to NS parameter on tape assignment control statement).	
	SI	Copy an SI coded format tape to mass storage, an I, or an SI binary tape file. The labeled or unlabeled input tape must be assigned as S format, with noise size of 8 for 7-track or 6 for 9-track tape (refer to NS parameter on tape assignment control statement). SI coded input tape is completed before EOI is encountered, the position of the input tape after the copy is indeterminate. This is because control words are used on the SI coded tape read via S format (EOF on an SI coded tape is a level 17_8 block terminator, whereas EOF on an S tape is a tape mark).	
TC=tc	Specifies the copy termination condition used in conjunction with N=copycnt. The termination condition can be specified as follows:		Copy to double EOF (TC=D or TC=EOD)
	F or EOF	When this TC value is set, the N keyword specifies the number of files to copy.	

Parameter		Description	Default
	I or EOI	This specifies a copy to the end of information. The N keyword is ignored.	
	D or EOD	When this TC value is set, the N keyword is the number of double EOFs to copy to.	
N=copycnt	Copy count used by the copy termination condition TC.		1
CC=charcnt	The character count which determines maximum block size (line length) in characters for an E or B format tape. This parameter can only be specified on an E or B format tape copy.		136 characters for E format; 150 characters for B format.
EL=erlimit	Error limit which sets the number of nonfatal errors allowed before abort. This includes parity errors and block-too-large errors which are returned by the tape subsystem after completing recovery procedures. It also includes illegal block format errors (invalid byte-count and/or unused bit count) for X format and SI coded format tapes. Error limit is ignored when generating an E or B format tape from mass storage, an I format, and an SI binary format file since control word read is not used. Error limit is likewise ignored if the input file device does not support control word read (terminals). In that case, any error aborts the job.		Zero
PO=p_1p_2	One or both of the following processing options (not separated by commas).		
	E	Input blocks with parity errors or block-too-large errors are processed (copied).	Error blocks are skipped.
	T	When generating a B or E format tape, blocks exceeding the maximum block size (refer to the CC parameter) are truncated. PO=T is illegal for other file conversions.	Lines exceeding the maximum line size are split into multiple blocks.
L=lfn_3	Name of an alternate output file to receive parity error messages when extended error processing is in effect (nonzero EL specified), in which case, the file name lfn_3 must not be the same as lfn_2.		OUTPUT

Example:

The following TCOPY statement combines order-dependent and order-independent parameters:

```
TCOPY(TAPE1,FILE2,E,CC=200,EL=12)
```

The input file TAPE1 is an E format tape (assigned as an S format tape). It has a maximum of 200 characters per line. The copy terminates when a double EOF is encountered (default). The output file FILE2 can be a mass storage file or an I or SI binary format tape. The error limit allows up to 12 nonfatal errors (parity/block-too-large), and the bad data is skipped (default) with informative error messages written to the file OUTPUT (default).

The TCOPY statement begins a copy operation at the current position of both files and continues until the copy termination condition is met or EOI is encountered. This termination condition can be a file count, double EOF count, or EOI. If the copy is terminated by a double EOF (for TC=EOD option), the second EOF is detected on lfn_1 but is not transferred to lfn_2. If lfn_1=lfn_2, the named file is read until the termination condition is satisfied or EOI is encountered. An SI coded tape can be positioned correctly only to EOI (refer to the F=SI parameter description).

If a copy specifies a file count TC=EOF, and EOI is encountered on the input file before the file count is satisfied, an additional EOF is written on the output file only if data or records have been transferred since the previous EOF was written (or since the beginning of the copy, if no EOFs have been encountered).

The EL or PO=E options provide extended error processing. This allows the processing or skipping of blocks with parity errors or block-too-large errors. If EL is set to a value greater than zero, a parity error or block-too-large error on the input tape generates the following message on the alternate output file.

```
PARITY/BLOCK TOO LARGE ERROR IN BLOCK n.
```

n is the decimal block count of the block in error. The block count for the first block to be copied is initially set to zero and is incremented by 1 for every block and every EOF processed. For X and SI coded formats, an illegal block format error (illegal byte count and/or unused bit count) produces the following message on the alternate output file.

```
ILLEGAL FORMAT IN BLOCK n.
```

When creating a B format tape from a mass storage, I, or SI binary format file, a block shorter than the noise size specified on the tape assignment statement is blank filled to the noise size. A noise block could also be generated when a block exceeding the maximum block size for the B format tape is split into multiple blocks. If the PO=T parameter is specified, blocks exceeding the maximum block size are truncated.

When creating an E format tape from a mass storage, I, or SI binary format file, blocks that exceed the maximum block size for the E format tape are split into multiple blocks. If a continuation block contains only the end-of-line indicator (zero word), the continuation block is discarded. If the PO=T parameter is specified, blocks exceeding the maximum block size are truncated (all continuation blocks are discarded).

TDUMP STATEMENT

The TDUMP control statement lists a file in octal and/or alphanumeric format. It dumps the entire file or the specified number of lines, records, or files. If more than one limit is set, the limit reached first overrides the others.

NOTE

TDUMP produces unpredictable results when dumping an S, L, or F format tape file. The user should use the COPY statement to convert the S, L, or F format tape file to a mass storage file or to an I or SI binary format tape file before attempting to dump the file using TDUMP.

The control statement format is:

TDUMP($p_1, p_2, \ldots, p_n$)

p_i Any of the following in any order:

$I=lfn_1$	1 to 7 alphanumeric characters naming the local file to be dumped (default is TAPE1).
$L=lfn_2$	1 to 7 alphanumeric characters naming the local file to which the output is written (default is OUTPUT). If lfn_2 is not a local file, TDUMP creates it. It does not rewind lfn_2 following the dump.
O	Octal dump only.
A	Alphanumeric dump only.
	If both O and A are specified, the last one overrides. If neither O nor A is specified, TDUMP lists both an octal and an alphanumeric dump.
R=rcount	Maximum decimal number of records to be dumped. If R is omitted or set to zero, the dump continues to EOI.

NOTE

The record count restarts at each EOF.

F=fcount	Maximum decimal number of files to be dumped. If F is omitted, the dump continues to EOI. If F=0, dump continues until an empty file (double EOF) or EOI is encountered.

N=lines Maximum decimal number of lines to be dumped. If N is omitted or set to zero, the dump continues to EOI. The blank line output with the end of record, end of file, end of information, and ABOVE LINE REPEATED messages is included in the line count.

NR Do not rewind file lfn_1 before dump (default is to rewind lfn_1).

Example:

Two lines, each containing the alphabet, were input to file X from a time-sharing terminal. File X was dumped to file Y producing the following output.

```
      - FILE DUMP -        TDUMP,I=X,L=Y.                                                    78/10/23. 08.05.51. PAGE    1.

F  1 R   1 W     0- 0102 0304 0506 0710 1112   1314 1516 1720 2122 2324   2526 2730 3132 0000 0000   0102 0304 0506 0710 1112
                     A B  C D  E F  G H  I J    K L  M N  O P  Q R  S T    U V  W X  Y Z              A B  C D  E F  G H  I J
F  1 R   1 W     4- 1314 1516 1720 2122 2324   2526 2730 3132 0000 0000
                     K L  M N  O P  Q R  S T    U V  W X  Y Z
-- END OF RECORD --

-- END OF INFORMATION --

-- END OF DUMP --
```

The prefix

F 1 R 1 W 0

means file 1, record 1, word 0. The zeros following each alphabet indicate the end of a terminal line.

UNLOAD STATEMENT

The UNLOAD control statement releases files assigned to the job and may release file space (depending on the file type).

The control statement formats are:

UNLOAD($lfn_1, lfn_2, \ldots, lfn_n$)

or

UNLOAD(*, $lfn_1, lfn_2, \ldots, lfn_n$)

The first format unloads the named files ($lfn_1, lfn_2, \ldots, lfn_n$). The second format unloads all files assigned to the job except the named files. If no files are named on the second format, the asterisk specification unloads all files assigned to the job.

The UNLOAD statement performs the same function as the RETURN control statement except as noted below. Refer to the description of the RETURN statement given earlier in this section to determine the operation performed for each file type.

The UNLOAD statement differs from the RETURN statement if the file being unloaded is a magnetic tape file or a direct access file residing on an auxiliary removable pack and the job requires more than one tape or pack resource concurrently. In this case, the UNLOAD statement does not decrease the number of tape/pack resources scheduled for the job with the RESOURC control statement.

For magnetic tape files, if the previous operation was a write, the UNLOAD statement causes the following operations to be performed.

- If the tape is ANSI labeled, the system writes a tape mark, an EOF1 label, and three tape marks and then unloads the tape.
- If the tape is unlabeled and the data format specified on the ASSIGN, LABEL, or REQUEST card is S, L, or F, the system writes four tape marks and then unloads the tape.
- If the tape is unlabeled and the data format is I or SI, the system writes a tape mark, an EOF1 label, and three tape marks and then unloads the tape.

Refer to Magnetic Tape Files in section 2, and Tape Management control statements in section 10 for further information about tape files and to appendix G for a description of an EOF1 label.

UNLOCK STATEMENT

The UNLOCK control statement rescinds the LOCK command and clears the write interlock bit for the specified file.

The control statement format is:

UNLOCK($lfn_1, lfn_2, \ldots, lfn_n$)

lfn_i Name(s) of local file(s)

The file must be a local file; if it is not, the following message is issued.

ILLEGAL FILE TYRE.

Library files cannot be unlocked.

VERIFY STATEMENT

The VERIFY routine performs a binary comparison of all data from the current position of the files specified. The comparison is meaningful if the files are within the range of compatible formats listed in table 1-7-2.

The control statement format is:

VERIFY(lfn_1, lfn_2, p_1, p_2, . . . , p_n)

lfn_1 Name of the first file; if this parameter is omitted, TAPE1 is assumed.

lfn_2 Name of the second file; if this parameter is omitted, TAPE2 is assumed.

p_i Any of the following in any order:

N=0	Verify terminates on the first empty file encountered on either file.
N=x	Verify x files; default is N=1.
N	Verify terminates when end of information is encountered on both files.
E=y	List the first y errors encountered on the comparison. If E is omitted, the system assumes E=100.
E	Same as E=0, no errors are listed.
L=lfn_3	List errors on file lfn_3. If L is omitted, the system assumes L=OUTPUT.
A	Abort after verify completed if errors occurred.
R	Rewind both files before and after the verify.
C	Coded file mode is set on both files. This is applicable only to S and L format tapes. If coded mode is set on an SI tape, the system aborts the job.
C1	Coded file mode is set on the first file only. This is applicable only to S and L format tapes. If coded mode is set on an SI tape, the system aborts the job.
C2	Coded file mode is set on the second file only. This is applicable only to S and L format tapes. If coded mode is set on an SI tape, the system aborts the job.
BS=bsize	Defines the maximum block size (PRU size) in central memory words for an S or L tape. This parameter is legal only for S and L tape verifies. The default for an S tape is 1000_8 words, and for an L tape, it is 2000_8 words.

Whenever words on lfn_1 and lfn_2 do not match, VERIFY lists the following.

- Record number
- Word number within the record
- Words from both files that do not match

If excess records are encountered on lfn_1 or lfn_2, the following message is listed.

n EXCESS RECORD(S) ON lfn.

n is the decimal number of excess records. The title line of the error list file contains the decimal number of the logical file being verified. If a nonstandard file (one in which an EOI or EOF is not preceded by an EOR) is compared with a standard file, VERIFY lists the following message.

r EOR MISSING ON lfn

r Record number in decimal

lfn Name of the nonstandard file

If EOI is encountered on one input file (lfn_1 or lfn_2) and there are still files remaining on the other input file, each excess file generates the following message.

n RECORD(S) IN EXCESS FILE m ON lfn.

n is the decimal number of excess records in logical file number m.

If errors are encountered, the following warning message is issued to the user's dayfile.

VERIFY ERRORS.

If any pair of lfn_1, lfn_2, and lfn_3 are identical, the following fatal message is issued.

FILE NAME CONFLICT.

If lfn_1 or lfn_2 did not exist prior to the verify, the following warning message is issued.

FILE NOT FOUND - lfn.

In a verify operation involving S, L, or F-format tapes, VERIFY first clears the extraneous data in the last word of each block (as specified by the byte count and the unused bit count) and then makes the comparison. On these formats, every block is considered a record (returns EOR status).

If a verification of an L or F format tape requires additional field length, VERIFY increases the field length as needed. If the field length requirement exceeds the user's maximum field length, the verify is aborted with the error message:

VERIFY FL ABOVE USER LIMIT.

The maximum block size for an L format tape is specified by the BS=keyword or its default. The maximum block size for an F format tape is calculated from the frame or character count specified on the control statement when the file is assigned.

A verify operation is not guaranteed when the logical structure of the two files is incompatible. Before VERIFY makes a comparison of such files, it issues the warning message:

FILE STRUCTURES NOT COMPATIBLE.

TABLE 1-7-2. COMPATIBLE FILE STRUCTURES FOR THE VERIFY STATEMENT

			OUTPUT MEDIA FORMAT					
				Tape Formats				
			Mass Storage	I	SI	S	L	F
INPUT MEDIA FORMAT	Mass Storage		Yes	Yes	Yes	No	No	No
	TAPE FORMATS	I	Yes	Yes	Yes	No	No	No
		SI	Yes	Yes	Yes	No	No	No
		S	No	No	No	Yes	No	No
		L	No	No	No	No	Yes	No
		F	No	No	No	No	No	Yes

NOTE

The No entries indicate that the logical structures of the files compared are incompatible. VERIFY may accept those combinations, but the results require the user to make a knowledgeable correlation of results with the format descriptions in section 10. In some cases, the verify of an incompatible pair may result in a VERIFY GOOD message; otherwise, a VERIFY ERRORS message is listed.

WRITEF STATEMENT

The WRITEF control statement directs the system to write a specified number of file marks on the named file.

The control statement format is:

WRITE(lfn,x)

lfn	Name of the file to be written on.
x	Number of filemarks to be written; if this parameter is omitted, the system assumes x=1.

If the last operation to the file was a write that did not end with the writing of an EOR or EOF, WRITEF writes a record mark before it writes the specified number of file marks. For all other cases, WRITEF writes the file marks without a preceding record mark.

WRITER STATEMENT

The WRITER control statement directs the system to write a specified number of empty records on the named file.

The control statement format is:

WRITER(lfn,x)

lfn	Name of the file to receive the empty records.
x	Number of empty records to be written; if this parameter is omitted, the system assumes x=1.

PERMANENT FILE CONTROL STATEMENTS 8

The permanent file control statements allow the user to utilize the permanent file system. † The control statements included in this category are:

APPEND	DEFINE	PERMIT	SAVE
ATTACH	GET	PURGALL	
CATLIST	OLD	PURGE	
CHANGE	PACKNAM	REPLACE	

The statements described in the following section allow the user to create permanent files (DEFINE) and make local files permanent (SAVE, REPLACE). These files can be accessed (ATTACH, OLD, GET), added to (APPEND), and released (PURGE, PURGALL). Requests are directed to a specified auxiliary device by the PACKNAM statement. Certain parameters can be changed with the CHANGE statement without attaching and redefining the file or retrieving and saving it.

Information on permanent files is obtained through the CATLIST statement. Part of that information is the permission status of the user as granted by another user by means of the PERMIT statement.

The following pages list options available on the control statements. Unless otherwise stated, the options described apply to all of the permanent file control statements. All file names must be 1 to 7 alphanumeric characters. For a detailed description of permanent file structure, refer to section 2. Errors encountered during permanent file control statement processing cause error messages to be issued to the user's dayfile. For a description of these messages, refer to appendix B.

† The batch user cannot access permanent files unless he has included a USER statement in the job deck.

Keyword	Option	Description
UN=	usernum	Alternate user number. This parameter is necessary only if the permanent file involved resides in another user's catalog. To be able to access other catalogs, the user must be granted explicit permission (refer to the PERMIT control statement), the file must be a semiprivate or public file, or the user must have automatic permission. A user has automatic permission to files in catalogs of other users if his user number contains asterisks, and all nonasterisk characters match the other user's user number. The UN keyword is used to establish alternate access validation (that is, PERMIT checking and catalog mode/category checking) even if the specified user number is the one under which the job is currently being run.
PW=	passwrd	The user has the option of specifying a 1- to 7-character password for a file. This password must be specified whenever alternate users access the file.
PW		The user has the added security of specifying a 1- to 7-character password for a file by including it as a single-line record in the INPUT file. This password must be specified whenever alternate users access the file.
CT=	ct	Permanent files fall into three categories which specify the method of access. This option must be selected when the file is saved, defined, or changed. The categories are: P or PRIVATE — Private files are available for access only by the originator or those to whom the originator has explicitly granted permission (refer to the PERMIT control statement). S or SPRIV — Semiprivate files are available for access by all users who know the file name, user number, and password. The system records in the originator's catalog the user number of each user who accessed the file, the number of accesses, and the date and time of the last access. PU or PUBLIC — Public files are available for access by all users who know the file name, user number, and password. The system records the number of times the file was accessed but does not record user numbers or the last access date and time.

Keyword	Option	Description
M=	m	Permanent file or user permission modes:
	E or EXECUTE	Allows the user to execute the file. To execute a file assigned to the user's job in EXECUTE mode, the file must either be in absolute format or in a relocatable format that can be loaded and executed via a file name call statement (such as LGO) which is not preceded by a loader control statement.
	R or READ	Allows the user to read and/or execute the file. Up to 63 users can access a file concurrently in R mode.
	RA or READAP	Allows the user to read and/or execute the file. For indirect access files, RA permission is the same as R permission. For direct access files, it allows the user to read and/or execute the file while another user is concurrently accessing the file in APPEND mode. Up to 63 users can access a file concurrently in RA mode.
	RM or READMD	Allows the user to read and/or execute the file. For indirect access files, RM permission is the same as R permission. For direct access files, it allows the user to read and/or execute the file while another user is concurrently accessing the file in MODIFY or APPEND mode. Up to 4095 users can access a file concurrently in RM mode.
	A or APPEND	Allows the user to read, execute, and/or append the file. Appending a file means adding data at the end of the file (EOI). Data within the original file boundaries cannot be changed.
	M or MODIFY	Allows the user to read, execute, append, and/or modify the file. For indirect access files, MODIFY permission is the same as APPEND permission. For direct access files, MODIFY permission means that the file can be changed or lengthened but not shortened.

Keyword	Option	Description
	W or WRITE	Allows the user to read, execute, append, modify, write, and/or purge the file. The file can be shortened, lengthened, or replaced.
	N or NULL	Removes permission previously granted via PERMIT control statements.

Special care should be taken when using READMD or READAP mode. Programs using access techniques (either the input/output macro or CYBER Record Manager) which do not expect concurrent updating of a file may get erroneous results if these modes are used.

CYBER Record Manager Advanced Access Methods (refer to the AAM Reference Manual) does not anticipate concurrent updating of a file by another user. Therefore, if a file has been attached in either READMD or READAP mode and these access methods are being used, a warning diagnostic message is issued stating that the file is bad when, in fact, it is not.

Keyword	Option	Description
SS=	subsystem	Specifies the time-sharing subsystem to be associated with the file. One of the following subsystems or its abbreviation may be specified on a SAVE or CHANGE control statement.

subsystem	Meaning	Abbreviation
BASIC	BASIC subsystem	BAS
BATCH	Batch subsystem	BAT
EXECUTE	Execute subsystem	EXE
FORTRAN	FORTRAN 5	FOR
FTNTS	FORTRAN Extended 4	FTN
NULL	NULL subsystem	NUL

In batch jobs, if the SS parameter is omitted or specified without a subsystem, the NULL subsystem is associated with the file.

In time-sharing jobs, if the SS parameter is specified without a subsystem, the currently active subsystem is associated with the file. If the SS parameter is omitted, the NULL subsystem is associated with the file, unless the file is the primary file. In that case, the current subsystem is associated with the file.

Keyword	Option	Description
PN=	packname	A 1- to 7-character pack name used in conjunction with the R keyword to identify the auxiliary device to be accessed in the permanent file request. This parameter is specified only when the file to be accessed resides on an auxiliary device. If the device is currently not available and the NA keyword was not specified, the following message is issued to the user's dayfile. DEVICE UNAVAILABLE, AT nnn. An auxiliary device is a mass storage device that supplements the normal family of permanent file devices. A RESOURC control statement must be included in any job that uses two or more disk packs concurrently.
S=	space	Specifies the amount of space in decimal PRUs desired for the direct access file. Refer to the DEFINE control statement.
R=	r	Specifies the type of device on which the permanent file resides or is to reside; r can be any of the following.

r	Device
DE	Extended Core Storage†
DIi	844-21 Disk Storage Subsystem ($1 \leq i \leq 8$) (half track)
DJi	844-4x Disk Storage Subsystem ($1 \leq i \leq 8$) (half track)
DKi	844-21 Disk Storage Subsystem ($1 \leq i \leq 8$) (full track)
DLi	844-4x Disk Storage Subsystem ($1 \leq i \leq 8$) (full track)
DMi	885 Disk Storage Subsystem ($1 \leq i \leq 3$) (half track)
DQi	885 Disk Storage Subsystem ($1 \leq i \leq 3$) (full track)
DP	Distributive Data Path to ECS†

The R keyword can be used in two ways.

- It can be used on the DEFINE control statement to specify the family device on which the direct access permanent file is to reside.

† The job must be of system origin or the user must be validated for system origin privileges.

Keyword	Option	Description
		• It can be used in conjunction with the PN and NA keywords on any permanent file control statement (including DEFINE) to identify the auxiliary device on which the permanent file resides or is to reside. R is required only if the desired device has a device type different from that of the default device type and the installation has defined the desired device as removable. If PN and NA are specified but R is not specified, the system default device type is used. If the specified device type cannot be recognized or does not exist in the system, the following message is issued to the user's dayfile. ILLEGAL DEVICE REQUEST, AT nnn.
NA		The NA keyword can be used in two ways. • Normally, if the user attempts to access a file that is interlocked or if an error occurs in an attempt to process the file, the system aborts the job. With the NA option, the user can bypass a job abort and continue processing. If lfn is busy and the NA option is specified on an ATTACH control statement, the system automatically suspends the job until the file becomes available. If NA is specified and an error other than pfn BUSY occurs in processing file lfn_i, the system issues the appropriate error message to the user's dayfile and then continues with file lfn_i+1. If the error occurred on the last file specified on the statement, the system continues with the next statement. • If the user requests an auxiliary device that is currently not available, the system aborts his job. The NA keyword enables him to bypass this abort and direct the system to make the desired device available.
ND		The ND keyword prevents releasing of the files assigned to the job upon processing of an OLD control statement.

Several files can be accessed with one control statement. A slash (/) is used to separate the files being accessed and the options described previously. The special options are order-independent and are indicated by the keywords described. If special options are specified on the control statement, they apply to all files that appear on the statement.

APPEND STATEMENT

The APPEND control statement allows the user to add supplementary information to an existing indirect access file.

The control statement format is:

APPEND(pfn, lfn_1, lfn_2, . . . , lfn_n/PW=passwrd, UN=usernum, PN=packname, R=r, NA)

pfn	Name of the indirect access permanent file to which the local files are to be appended.
lfn_i	Name(s) of local file(s) to be appended to pfn.

The logical structure of the two files is retained; that is, EORs and EOFs are appended as well as data. If the file is appended to a file in an alternate user's catalog, a password must be supplied if one is required.

ATTACH STATEMENT

The ATTACH control statement allows a user to access a direct access file.

The control statement format is:

ATTACH(lfn_1=pfn_1, lfn_2=pfn_2, . . . , lfn_n=pfn_n/UN=usernum, PW=passwrd, M=m, PN=packname, R=r, NA)

lfn_i	Local file name given to the direct access file while it is attached to the user's job. A working copy is not generated since user access is made directly to the permanent file. Thus, lfn_i is used only when it is desirable to reference the attached file by a name other than its permanent file name, pfn_i. The local file name is returned to the system if it is already present when this statement is issued, even if an error is encountered in processing the statement.
pfn_i	Name of direct access file to be attached. If pfn_i is omitted, the system assumes pfn_i=lfn_i.
m	File or user permission mode, where m can be W, M, A, E, R, RM, or RA. If m is omitted, the system assumes m is R. This option must be specified by all users, including the originator, if the file is to be modified or new information is to be added to the file. If pfn_i is attached in W mode, the date is recorded as last modification date even if the file was not altered.

A read/write interlock controls multiple access of a direct access file. The main purpose of this interlock is to ensure that only one user at a time writes on the file; however, it is possible for several users to read a file simultaneously.

Table 1-8-1 gives combinations of multiple access. The left column specifies the current access status of the file, and the top row indicates the type of access a user is requesting on an ATTACH statement with the M parameter. The entries in the table are the access modes actually granted. The access a user is granted is contingent on having been permitted that mode of access by the creator of the file.

TABLE 1-8-1. COMBINATIONS OF MULTIPLE ACCESS

Current Access	Access Requested						
Free	W	M	A	R	RM	RA	E
W	Busy	Busy	Busy	Busy	Busy	Busy	Busy
M	Busy	Busy	Busy	Busy	M/R	Busy	Busy
A	Busy	Busy	Busy	Busy	A/R	A/R	Busy
R	Busy	Busy	Busy	R	R	R	R
RM	Busy	M/R	A/R	R	R	R	R
RA	Busy	Busy	A/R	R	R	R	R
E	Busy	Busy	Busy	R	R	R	R

NOTES

W, M, A, R, RM, RA, and E have the values described under the M= keyword.

Busy indicates the requested access is not allowed while the current access is in effect.

A/R is the access condition in which one user has attached the file in append mode, and one or more other users have attached it in read mode.

M/R is the access condition in which one user has attached the file in modify mode, and one or more other users have attached it in read mode.

If a file is to be accessed by alternate users, it should be returned as soon as possible.

If an auxiliary device has been previously specified by a PACKNAM statement, the system attempts to attach pfn_i from the auxiliary device rather than the normal system devices.

CATLIST STATEMENT

The CATLIST control statement lists information about the user's permanent files or those permanent files he can access in the catalogs of alternate users.

The control statement format is:

CATLIST(LO=p,FN=pfn,UN=usernum,PN=packname,R=r,L=lfn,NA,DN=dn)

LO=p One of the following list options (default is 0):

F Lists pertinent information about each file in the user's catalog. The final two lines give the number of indirect access files, the number of direct access files, and the total PRUs used by each type.

If an alternate user number is specified (UN option), the user obtains a listing of all files that he can access in the alternate user's catalog. The password for files in an alternate user's catalog is not included in the listing. The password to files in an alternate user's catalog must be obtained directly from that user.

FP Lists the permission information recorded for each alternate user of a specified file in the user's catalog. This option requires that a file name be specified (FN option). If an alternate user number is specified (UN option), only the permission information for that user of the specified file is listed.

The user numbers listed include those that have been granted explicit permission to the file (private file only) and those that have accessed the file because of implicit permission (semiprivate files only).† An asterisk (*) follows the user number/permission mode if explicit permission has been granted this user.

0 (zero) Lists alphabetically by column the names of the indirect access files and the names of the direct access files in the user's catalog. If an alternate user number is specified (UN option), the user obtains only the names of the files that he can access in the alternate user's catalog. If no LO keyword is specified, the system assumes this value.

An asterisk (*) preceding a file name indicates an error status is set in the catalog entry for the file. The cause of the error may be one of the following.

- EOI was altered during mass storage recovery.
- BOI/EOI verification error.
- Error in data and/or permit entries.

To clear an error status flag, refer to the CHANGE statement in this section.

†User numbers are not recorded for accesses to public files.

P — Selects a short list that indicates only the user numbers of alternate users who have access to the specified private or semiprivate file. This option requires that a file name be specified (FN option).

FN=pfn — Permanent file name. If pfn contains no asterisks, this option requests catalog information only for the permanent file pfn. This parameter is required when listing permit information (the LO=FP and LO=P list options). If a short list option is selected (LO=0 or LO=P), the message

pfn FOUND, AT nnn.

is issued if the file (or user number) is located. The message

pfn NOT FOUND, AT nnn.

is issued if the file (or user number) is not located.

If pfn contains one or more asterisks, CATLIST lists catalog information for the subset of files whose names contain the same letters in the same positions as specified in pfn. For example, FN=***OPL lists all 6-character file names ending in OPL. FN=M****** lists all files whose names start with the letter M. The asterisk is invalid when listing permit information with the LO=FP or LO=P list options.

UN=usernum — User number. This parameter has two purposes.

- For LO=F and LO=0. Indicates the alternate catalog for which the user desires catalog information.
- For LO=FP and LO=P. Indicates the permission information recorded for the specified alternate user.

PN=packname — This parameter specifies an auxiliary device that contains catalog information for all users with files on that device. The PN keyword must be specified if the user wishes to obtain the following information from his catalog on the specified auxiliary device.

- Pertinent information about each file (LO=F).
- Only the name of each file (LO=0).
- Permission information for each alternate user that has accessed a specific file (LO=FP).
- Only the user number of each alternate user that has accessed a specific file (LO=P).

The PN parameter can also be specified to allow alternate users to obtain a list of files they can access on the auxiliary device, as well as pertinent information about each file.

R=r	Device type on which permanent file catalog resides. Used in conjunction with the PN and NA parameters. Refer to R parameter description at beginning of this section.
L=lfn	Output file name. This is the name of a local file to which the CATLIST information is written. If this parameter is omitted, the system assumes L=OUTPUT. If lfn exists and is positioned at BOI, the contents of that file is purged before the CATLIST information is written. However, if lfn exists and is positioned at EOI, the CATLIST information is appended to the file as a new logical record.
NA	No abort option. CATLIST continues processing if errors are encountered during processing.
DN=dn	Device number (0 through 77_8). List file residing on specified device number dn.

If no entries are present in the specified catalog, the message

EMPTY CATALOG.

is issued to the user's dayfile.

Example:

A user entered the following statement

CATLIST.

and received the following listing of his permanent files.

```
CATALOG OF USERNUM              FM/FAMNAME 79/05/14. 10.37.26.

INDIRECT ACCESS FILE(S)

 CONVER    FJOB      LFILE     PROC12    PROC13    TESTA     ZZZDUMP
 DJOB      FORT

DIRECT ACCESS FILE(S)

 DATAB     LIB5      TESTLIB   TEST2

       9 INDIRECT ACCESS FILE(S),   TOTAL PRUS =        19.
       4 DIRECT ACCESS FILE(S),     TOTAL PRUS =         4.
```

The heading gives the user number, the device family name, and the date and time. If the PN=packname parameter was specified, the family name in the heading is replaced by PN/packname.

CHANGE STATEMENT

The CHANGE control statement allows the creator of a direct or indirect access permanent file to change one or more of its characteristics without assigning the file to his job. A direct access file need not be attached and redefined; an indirect access file need not be retrieved and replaced.

The control statement format is:

CHANGE(nfn=ofn/CT=ct,M=m,PW=passwrd,SS=subsystem,PN=packname,R=r,NA,CE)

The full descriptions of the statement parameters are given at the beginning of this section.

nfn	New permanent file name.
ofn	Old permanent file name. If no name change is desired, only ofn is specified.
CT=ct	New access category for the file (private, semiprivate, or public).
M=m	New alternate user permission mode.
PW=password	New password. If PW=0 is specified, CHANGE clears the old password without setting a new password.
SS=subsystem	New time-sharing subsystem to be associated with the file.
PN=packname	Auxiliary pack on which the file resides. This parameter cannot specify a new file residence.
R=r	Device type on which the file resides. This parameter cannot specify a new file residence.
NA	If the requested auxiliary pack is not available, the job is suspended until the pack becomes available.
CE	Clear file error code. For further information, refer to section 5 in volume 2.

CHANGE also updates the last modification date and last access date for the file.

DEFINE STATEMENT

The DEFINE control statement allows the user to define direct access permanent files.

The control statement format is:

DEFINE(lfn_1=pfn_1,lfn_2=pfn_2,...,lfn_n=pfn_n/PW=passwrd,CT=ct,M=m,R=r,
S=space,PN=packname,NA)

The full descriptions of these parameters are given at the beginning of this section.

lfn_i	If DEFINE creates an empty direct access permanent file, lfn_i is specified only if the user desires to reference the file by a name other than its permanent file name. If DEFINE defines an existing local file as a direct access file, lfn_i is the name of the local file. Also, if lfn_i exists, its position is not altered.
pfn_i	Permanent file name. If pfn_i is omitted, the system assumes lfn_i=pfn_i.
PW=passwrd	Password required to access the defined file.
CT=ct	Access category of the defined file (private, semiprivate, or public).

M=m	Alternate user permission mode. This does not affect the current mode. After defining a file, the user is always in write mode.
R=r	Type of device on which the permanent file is to reside. The device must be a permanent file mass storage device on which direct access files are allowed.
S=space	Number of PRUs requested for the file.
PN=packnam	Name of the auxiliary pack on which the direct access file is to reside.
NA	If the requested auxiliary pack is not available, the job is suspended until the pack becomes available.

The user can either create an empty permanent file or define an existing local file as a direct access file. If the user releases the file and wishes to access it at some time in the future, the ATTACH control statement must be included.

If lfn_i does not exist, the device on which pfn_i resides depends on the r and space parameters.

r	space	Residency
Specified	Not specified	The file resides on the device of type r with the most space available.
Specified	Specified	The file resides on the device of type r with the most space available, provided that device has as many PRUs available as specified by the space parameter.
Not specified	Specified	The file resides on the device with the most space available, provided that device has as many PRUs available as specified by the space parameter.
Not specified	Not specified	The file resides on the device with the most space available.

If an auxiliary device has been previously specified by a PACKNAM statement, pfn_i resides on that auxiliary device rather than a system device.

If the optional parameters are omitted, the system assumes the following values.

Keyword	Default
PW	None
CT	PRIVATE
M	WRITE
PN	None

If the S option is selected and no device has the specified amount of space available, the request is aborted and the following message is issued to the user's dayfile.

PRUS REQUESTED NOT AVAILABLE, AT nnn.

Unused space is not guaranteed to be available if the user attempts to expand the file at a later time.

If lfn_i already exists on a device other than that specified by r, or an illegal device is specified, the system issues the following message to the user's dayfile.

DIRECT ACCESS DEVICE ERROR, AT nnn.

GET STATEMENT

The GET control statement enables the user to retrieve a copy of file pfn_i for use as a local file.

The control statement format is:

GET(lfn_1=pfn_1, lfn_2=pfn_2, . . . , lfn_n=pfn_n/UN=usernum, PW=passwrd, PN=packname, R=r, NA)

lfn_i	Local file name given the file while in use.
pfn_i	Permanent file name; if pfn_i is omitted, lfn_i=pfn_i.

If the request is made with no parameters specified, the user's primary file is assumed.

Each pfn specified must be an indirect access file. File lfn_i is returned to the system if it is present before this command is issued even if an error is encountered in processing the command. The new file is rewound. No interlock is provided to prevent other users from obtaining working copies of the same file simultaneously. If the name of the user's current primary file is specified as an lfn, the corresponding pfn is made the new primary file and any subsystem associated with it becomes the user's new current time-sharing subsystem (refer to the IAF Reference Manual or the Time-Sharing User's Reference Manual).

If the request is for a file in another user's catalog (UN option specified), the permission mode must allow the user to read the file.

If an auxiliary device has been previously specified by a PACKNAM statement, the system attempts to retrieve the copy of pfn_i from the auxiliary device rather than the normal system devices.

OLD STATEMENT

The OLD control statement retrieves a copy of a permanent file and makes it the primary file.

The control statement format is:

OLD(lfn=pfn/UN=usernum, PW=passwrd, PN=packname, R=r,NA,ND)

lfn	Local file name given the file while in use.
pfn	Permanent file name. If pfn is omitted, lfn=pfn.

The OLD statement performs the same operation as the GET statement and additionally makes lfn the primary file. Any currently existing primary file is released. All files assigned to the job are released unless the ND parameter is specified.

If an auxiliary device has been specified previously by a PACKNAM statement, the system attempts to retrieve the copy of pfn from the auxiliary device rather than the normal system devices.

Refer to the note in PRIMARY Statement in section 7 for use of the primary file type.

PACKNAM STATEMENT

The PACKNAM control statement directs subsequent permanent file requests to the specified auxiliary device.

The control statement format is:

PACKNAM(PN=packname)

or

PACKNAM(packname)

packname	A 1- to 7-character name used to identify the auxiliary device to be accessed in subsequent permanent file requests.

PACKNAM allows the user to omit the PN keyword from requests for files that reside on the specified device. However, if permanent files on another auxiliary device are to be accessed, the PN keyword can be specified in the request or another PACKNAM request can be issued. Refer to Mass Storage File Residence in section 2 for information concerning auxiliary permanent file devices.

The user cannot access permanent files residing on the normal system devices while the PACKNAM request is in effect. To access these files, he must include a PACKNAM statement in either of the following formats.

PACKNAM.

or

PACKNAM(PN=0)

PERMIT STATEMENT

The PERMIT control statement allows a user to explicitly permit another user to access a private or semiprivate file in his permanent file catalog.

The control statement format is:

PERMIT(pfn, $usernum_1$=m_1, $usernum_2$=m_2, . . . , $usernum_n$=m_n/PN=packname, R=r, NA)

pfn	Permanent file name.
$usernum_i$	User number to be permitted access to pfn.
m_i	Permitted mode of access. If m_i is omitted, the system assumes mode R. (The access modes are defined at the beginning of this section.)

If pfn is a public file, the following message is issued.

PFM ILLEGAL REQUEST, AT nnn.

PURGALL STATEMENT

The PURGALL control statement purges all permanent files in the user's catalog that satisfy the criteria specified by the parameters.

The control statement format is:

PURGALL(CT=ct, AD=ad, MD=md, CD=cd, DN=dn, TY=ty, TM=tm, PN=packname, R=r, NA)

ct	File category.
ad	Last access date; format of date is yymmdd.
md	Last modification date; format is yymmdd.
cd	Creation date; format is yymmdd.
dn	Device number (0 through 77_8). The device number is assigned during system configuration time when the device is defined. It uniquely identifies a device within a family. †
ty	File type: I or INDIR — Purge all indirect access files D or DIRECT — Purge all direct access files A or ALL — Purge all files If this parameter is omitted but other parameters are specified, the system assumes ty is ALL. If no other parameters are specified and the user wishes to purge all files, he must specify TY=A.
tm	Time of day on the date specified by ad, md, or cd parameter. The time of day is expressed in the format hhmmss.

† Refer to section 2 for further information about families of permanent file devices.

packname	Name of auxiliary device on which the files to be purged reside. The PN option cannot be selected if a device number was specified.
r	Type of auxiliary device on which the files to be purged reside. The R option cannot be selected if a device number was specified.

The AD, MD, and CD keywords are used to purge any files whose last access, last modification, or creation occurred before the specified date. To purge all files in his catalog, the user must enter:

```
PURGALL(TY=A)
```

CT, DN, TY, TM, and either AD, MD, or CD may be entered simultaneously.

PURGE STATEMENT

The PURGE control statement names files to be removed from the permanent file device.

The control statement format is:

PURGE(pfn_1, pfn_2, . . . , pfn_n/UN=usernum, PW=passwrd, PN=packname, R=r, NA)

pfn_i	Permanent file name.

If the request is made with no parameters specified, the user's primary file is assumed.

When a PURGE command is issued for a direct access file which is not being used, the file is purged and the permanent file catalog is altered accordingly. If the direct access file is in use, the catalog is altered to reflect purging of the permanent file but the actual file is not purged until the last user returns it.

To purge a file in an alternate user's catalog, one of the following must be true.

- The file is private and the user has write permission.
- The file is public with write mode.
- The file is semiprivate and the user has write permission.
- The file is semiprivate with write mode and the user has not explicitly been permitted access to the file.

If pfn_i does not exist, the following message is issued.

```
pfn NOT FOUND, AT nnn.
```

REPLACE STATEMENT

The REPLACE control statement enables the user to place a copy of a local file in the permanent file system as an indirect access file.

The control statement format is:

REPLACE(lfn_1=pfn_1, lfn_2=pfn_2, . . . , lfn_n=pfn_n/UN=usernum, PW=passwrd, PN=packname, R=r,)

lfn_i	Local file name.
pfn_i	Permanent file name. If pfn_i is omitted, lfn_i=pfn_i.

If the request is made with no parameters specified, the user's primary file is assumed.

If pfn_i already exists, it is purged and replaced by the new file. The new file is in the same category as the file it replaced. If pfn_i does not exist, the new file is saved as a private file. The specified local files are rewound before and after the replace operation. Permission information and alternate user access data for the file are not lost when a file is replaced.

A user who has been granted write permission to another user's file can replace that file only if he is validated to create indirect access permanent files (refer to LIMITS control statement in section 6).

SAVE STATEMENT

The SAVE control statement allows the user to retain a copy of a local file as an indirect access file.

The control statement format is:

SAVE(lfn_1=pfn_1. lfn_2=pfn_2, . . . , lfn_n=pfn_n/PW=passwrd, CT=ct, M=m, SS=subsystem, PN=packname, R=r, NA)

lfn_i	Local file name.
pfn_i	Permanent file name. If pfn_i is omitted, the system assumes lfn_i=pfn_i.

If the request is made with no parameters specified, the user's primary file is assumed. If the name of the user's current primary file is specified as an lfn, the user's current subsystem is stored in the file's catalog entry.

The specified local files are rewound when the save operation is completed. If the optional parameters are omitted, the system assumes the following values.

Parameter	Default
PW=passwrd	No password.
CT=ct	Private access category.
M=m	WRITE alternate user permission.

Parameter	Default
SS=subsystem	NULL subsystem, unless lfn is the primary file in a time-sharing job. The subsystem is then set to the currently active subsystem.
PN=packname	The file does not reside on an auxiliary pack.

The full descriptions of the statement parameters are given at the beginning of this section.

If an auxiliary device has been previously specified by a PACKNAM statement, the system saves pfn_i on the auxiliary device rather than a normal system device. If pfn_i already exists in the user's catalog, the following message is issued.

pfn ALREADY PERMANENT, AT nnn.

ERROR CONDITIONS

Table 1-8-2 specifies the action PFM takes if it detects an error while reading mass storage. The symbols used in the table designate the response PFM makes and are defined as follows:

Symbol	Description
DTE	DATA TRANSFER ERROR.
EOI	Processing continues as if an EOI was encountered.
MSE	MASS STORAGE ERROR.
FNF	pfn NOT FOUND.
DAF	DIRECT ACCESS FILE ERROR.
FLE	FILE LENGTH ERROR.

TABLE 1-8-2. PERMANENT FILE ERROR CONDITIONS

Activity	Command								
	SAVE	GET	PURGE	CATLIST	PERMIT	REPLACE	APPEND	DEFINE	ATTACH
Device-to-device transfer (valid sector)	DTE	DTE				DTE	DTE		
Device-to-device transfer (no valid sector)	EOI†	EOI†				EOI†	EOI†		
Reading PF catalog	MSE	FNF	FNF	EOI	FNF	MSE	FNF††	MSE	FNF
Device-to-device transfer of original file (valid sector)							DTE		
Device-to-device transfer of original file (no valid sector)							EOI†		
Reading a system sector			DAF					DAF	DAF
Reading permit information		FNF	FNF	EOI		FNF	FNF		FNF
Reading permit information for update		MSE			MSE	MSE	MSE		MSE

† Unless the error occurred while the last sector was being read, a FILE LENGTH ERROR message is issued.
†† If the error occurred on a reentrant search of the PF catalog, a MASS STORAGE ERROR message is issued.

LOAD/DUMP CENTRAL MEMORY UTILITY CONTROL STATEMENTS

9

The load/dump central memory utility control statements allow the user to transfer information that resides in his job field length to a peripheral device or to transfer information from that device into central memory. The following statements are included in this category.

DMP	DMPECS	PBC
DMD	LBC	RBR
DMDECS	LOC	WBR

NOTE

For information concerning security restrictions associated with the use of these control statements, refer to Security Control in section 3.

The DMP and DMD control statements dump central memory in octal representation and/or display code equivalences. Likewise, the DMDECS and DMPECS control statements dump ECS memory. These statements are particularly helpful in creating dumps for debugging purposes. (Refer to Debugging Aids in section 12.) Other transfers of data from central memory use the PBC statement which dumps a binary record to PUNCHB and the WBR statement which writes a binary record on a specified file.

Data is loaded to central memory by the LBC, LOC, and RBR statements. The LBC control statement is useful in loading binary data in an unknown format. All numeric parameters may be expressed in octal (postradix is B) or decimal (postradix is D) notation. If no radix is specified, octal is assumed.

DMP STATEMENT

The DMP control statement requests a dump on file OUTPUT of central memory in four words per line.

The control statement format is:

```
DMP(fwa,lwa)
    or
DMP(lwa)
    or
DMP.
```

fwa — First word address of memory to be dumped; fwa is relative to RA. If fwa is absent, dump mode depends on the presence or absence of lwa. If fwa is greater than the user's field length, fwa is set at the field length minus 10_8. If fwa is greater than or equal to 400000_8, the first dump address is fwa minus 400000_8, memory from the first dump address through lwa is dumped, and the job is aborted.

lwa Last word address plus 1 of memory to be dumped; lwa is relative to RA. If lwa alone is present, DMP assumes fwa equals 0. If neither fwa nor lwa is present, DMP dumps the exchange package and 40_8 locations before and after the program address register in the exchange package. If lwa is greater than the user's field length, the dump stops at the end of the field length.

If either fwa or lwa is nonnumeric, DMP dumps the exchange package and 40_8 locations before and after the program address register in the exchange package. If both fwa and lwa are greater than the user's field length, the last 10_8 words of the user's field length are dumped. If fwa equals lwa, the 10_8 words beginning at fwa are dumped. If fwa is greater than lwa, DMP issues an error message and terminates the job step.

The user must not place another control statement (other than DMP, DMD, DMPECS, DMDECS, or EXIT) between the program to be dumped and the DMP statement.

DMP suppresses duplicate lines and then issues the following output message.

DUPLICATED LINES.

In time-sharing jobs, DMP is effective only within procedure files. A dump from a terminal is formatted for 72-column output and written on local file ZZZDUMP. DMP displays an informative message at the terminal.

DMD STATEMENT

The DMD control statement requests a dump similar to that of the DMP statement but adds the display code equivalences to the right of the octal representations.

The control statement format is:

```
DMD(fwa,lwa)
    or
DMD(lwa)
    or
DMD.
```

fwa First word address of memory to be dumped; fwa is relative to RA. If fwa is absent, dump mode depends on the presence or absence of lwa.

lwa Last word address plus 1 of memory to be dumped; lwa is relative to RA. If lwa alone is present, DMD assumes fwa is 0. If neither fwa nor lwa is present, DMD dumps the exchange package and 40_8 locations before and after the program address in the exchange package.

The DMD statement can be used from a time-sharing terminal only in a procedure file and only after OUTPUT is assigned to mass storage, as in the following example.

```
.PROC,PROCA.
ASSIGN(MS,OUTPUT)
FTN(I=PROG)
LGO.
EXIT.
DMD(0,100)
ROUTE(OUTPUT)
```

DMPECS STATEMENT

The DMPECS control statement dumps the contents of an ECS field length on file OUTPUT or a user-specified file. The dump is four words per line. If lines are duplicated, they are suppressed and the following notation is issued to the output file.

DUPLICATED LINES.

A DMPECS statement within a time-sharing job copies the contents of the ECS field length to the local file ZZZDUMP and displays a message at the terminal informing the user of the dump.

The control statement formats are:

DMPECS(fwa,lwa)
DMPECS(lwa)
DMPECS(fwa,lwa,f,lfn)

fwa	First word address of ECS memory to be dumped; fwa is relative to the reference address of the field in ECS being used by the job (RAE).
lwa	Last word of ECS memory to be dumped; lwa is relative to RAE.
f	Print format (included for compatibility with NOS/BE).
lfn	File to dump to.

If the first format is used, the field in ECS memory defined by fwa and lwa is dumped to the file OUTPUT. Display code equivalences do not appear.

If the second format is used, DMPECS assumes fwa is 0. Display code equivalences do not appear.

If the third format is used, the specified field in ECS is dumped to lfn. The parameter f is ignored. Display code equivalences appear to the right of the octal representations, the same as the DMDECS control statement.

The DMPECS statement must immediately follow a program to be dumped, except that another DMDECS or DMPECS, DMP, DMD, or EXIT may intervene.

Dumping always stops at the field length in ECS (FLE) if lwa is greater than FLE. If either fwa or lwa is nonnumeric, the following error message is issued to the user's dayfile.

ARGUMENT ERROR.

If fwa is greater than FLE, fwa is set to FLE-10. If both fwa and lwa are greater than FLE, fwa is set to FLE-10 and lwa is set to FLE. If fwa is greater than lwa, the system issues the following message to the user's dayfile.

FWA .GE. LWA+1.

If neither fwa nor lwa is specified, the following message is issued to the user's dayfile.

ILLEGAL REQUEST.

If no ECS field length exists for the user, the following message is issued to the user's dayfile.

NO ECS.

DMDECS STATEMENT

The DMDECS control statement requests a dump of ECS memory on file OUTPUT. The dump is four words per line with display code equivalences to the right of the octal representations. If lines are duplicated, they are suppressed, and the following notation is issued to the output file.

```
DUPLICATED LINES.
```

The control statement formats are:

```
DMDECS(fwa,lwa)
DMDECS(lwa)
```

fwa — First word address of ECS memory to be dumped; fwa is relative to the reference address of the field in ECS being used by the job (RAE). If fwa is absent, DMDECS assumes fwa is 0.

lwa — Last word address of ECS memory to be dumped; lwa is relative to RAE.

The DMDECS statement must immediately follow a program to be dumped, except that another DMDECS or a DMPECS, DMP, DMD, or EXIT may intervene.

Dumping always stops at the field length in ECS (FLE) if lwa is greater than FLE. If either fwa or lwa is nonnumeric, the following error message is issued to the user's dayfile.

```
ARGUMENT ERROR.
```

If fwa is greater than FLE, fwa is set to FLE-10. If both fwa and lwa are greater than FLE, fwa is set to FLE-10 and lwa is set to FLE. If fwa is greater than lwa, the system issues the following message to the user's dayfile.

```
FWA .GE. LWA+1.
```

If neither fwa nor lwa is specified, the following message is issued to the user's dayfile.

```
ILLEGAL REQUEST.
```

If no ECS field length exists for the user, the following message is issued to the user's dayfile.

```
NO ECS.
```

The DMDECS statement can be used from a time-sharing terminal only in a procedure file and only after OUTPUT is assigned to mass storage, as in the following example.

```
.PROC,PROCB.
ASSIGN(MS,OUTPUT)
FTN(I=PROG)
LGO.
EXIT.
DMDECS(0,100)
ROUTE(OUTPUT)
```

LBC STATEMENT

The LBC control statement is intended for loading binary data of unknown format.

The control statement format is:

```
LBC(addr)
```

addr — Address relative to RA at which binary load begins; if addr is omitted, 0 (RA) is assumed.

LBC reads only one record from file INPUT. The user must make an LBC call for each record of data to be loaded. If addr is specified in the program call, binary data is loaded beginning at that address; otherwise, loading begins at the reference address (RA).

LOC STATEMENT

The LOC control statement reads octal line images from file INPUT and enters them in the user's CM field length.

The control statement format is:

```
LOC(fwa,lwa)
    or
LOC(lwa)
    or
LOC.
```

fwa — First word address of an area to clear (zero) before loading correction statements. If fwa is absent, LOC assumes 0.

lwa — Last word address plus 1 of the area to be cleared. If lwa is absent, LOC assumes 0.

To process the LOC statement, the system reads correction statement images from the current INPUT record. A correction statement consists of an octal address and a data field. The address field specifies the location to be corrected, and the data field contains the data to be placed in that location. Both fields may start at any column as long as the address precedes the data. The address field consists of a one- to six-digit address. If it is 5 characters or less, it is separated from the data field by a nonoctal character (for example, a blank). If it is 6 characters, no separator is required.

The data field consists of 1 to 20 octal characters. If it is less than 20 characters, it is terminated by a nonblank, nonoctal character and is stored right-justified. If it is 20 characters, no terminator is required. Embedded blanks in the data field are ignored.

If both fwa and lwa are specified and both are nonzero, storage is cleared from fwa to lwa, and the octal line images are loaded at the specified addresses. If the current INPUT record is empty, LOC clears the indicated area of memory.

PBC STATEMENT

The PBC routine writes one record from the specified area of CM to file PUNCHB.

The control statement format is:

```
PBC(fwa,lwa)
    or
PBC(lwa)
    or
PBC.
```

fwa — Address relative to RA at which the binary deck begins; if this parameter is omitted, the PBC operation depends upon the presence or absence of lwa.

lwa — Last word address plus 1 of the binary deck. If lwa alone is present, PBC assumes that fwa is RA. If lwa is fwa, and a nonzero value is specified, PBC adds 10_8 to lwa. If fwa and lwa is 0 or are omitted, RA contains lwa in the lower 18 bits. If the upper 12 bits of RA are 7700_8, lwa is the lower 18 bits of the location following the prefix (77) table plus the length of the prefix table.

CM is not altered by PBC.

RBR STATEMENT

The RBR routine loads one binary record from a specified file.

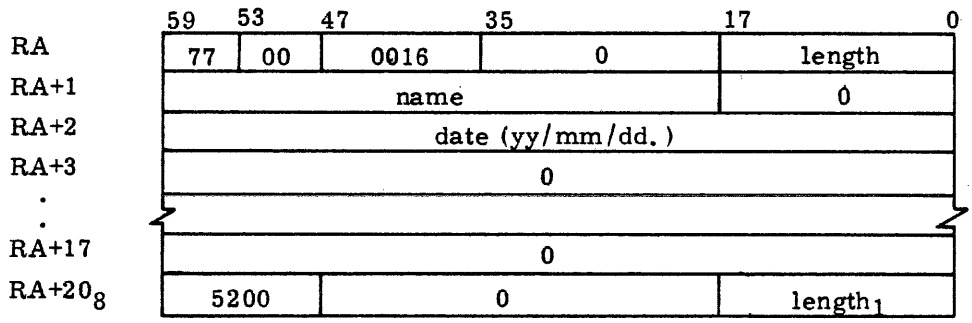
The control statement format is:

RBR(n, name)

n — n is used in constructing the name of the file containing the binary record to be read. If n is less than 4 characters and is numeric, TAPEn is the file name. If n contains a nonnumeric character or is 4 or more characters long, n itself is used as the file name. If n is absent, TAPE is the file name.

name — A 1- to 7-character name used in a record prefix.

The RBR routine loads one binary record from the specified file into central memory starting at RA. If the name parameter is included, a record prefix is placed in central memory starting at RA. The record itself follows. The following is the format of the record prefix.



length Record length including the prefix

length$_1$ Record length minus words RA through RA+17$_8$

If the record is too long for available memory, memory is filled, excess data is skipped, and the following message is issued to the user's dayfile.

RECORD TOO LONG.

WBR STATEMENT

The WBR routine writes a binary record from CM to a file at its current position.

The control statement format is:

WBR(n, rl)

n — n is used in constructing the name of the file on which the binary record is to be written. If n is less than 4 characters and is numeric, TAPEn is the file name. If n contains a nonnumeric character or is 4 or more characters long, n itself is used as the file name. If n is absent, TAPE is the file name.

rl — Record length in words. If rl is 0 or absent, the length is taken from the lower 18 bits of RA.

WBR begins writing from RA.

TAPE MANAGEMENT 10

This section describes control statements used with magnetic tape files.† For additional information on NOS magnetic tape files, consult the glossary for definitions of terms; Magnetic Tape Files in section 2 for descriptions of tape labels and data formats; and appendix G for tape label formats. Section 6 describes the RESOURC statement required in jobs that use more than one tape or removable auxiliary pack concurrently.

NOTE

The term file as used in this section may refer to a multifile file. Refer to table 1-2-1 for the EOR and EOF marks for tape files.

The control statements described in this section are:

ASSIGN	Assigns a local file to a tape unit† (system origin jobs or jobs with system origin privileges only). Section 7 describes the ASSIGN statement for nontape files.
BLANK	Blank labels a tape and may restrict access to the labeled tape.
LABEL	Assigns a local file to a magnetic tape,† creates and verifies tape labels, and creates and accesses multifile set tapes.
LISTLB	Lists tape labels.
REQUEST	Assigns a local file to a magnetic tape device.†
VSN	Associates a file name with one or more VSNs for later assignment by a LABEL or REQUEST statement.

TAPE ASSIGNMENT

Whenever a tape is mounted, the system checks for labels. If the tape is labeled, the system records the volume serial number (VSN) read from the VOL1 label and the equipment on which the tape is mounted. When a tape assignment is requested by a LABEL or REQUEST statement specifying an lfn and a VSN (or an lfn that has been named in a previous VSN statement), the system compares the VSN with the VSNs read from mounted tapes. If a match is found, the system automatically assigns the tape to the requesting job, provided a deadlock would not occur.†† If the tape is not mounted, the system rolls out the job until a tape with the requested VSN is mounted.

†If the user does not specify a VSN parameter or an MT or NT parameter on the tape assignment statement, the operator can assign any device to the file.

††Refer to the RESOURC statement in section 6.

For a mounted, unlabeled tape, the operator enters a command specifying the requested VSN. The system can then assign the tape. A VSN which contains nonalphanumeric characters should not be specified in a request for an unlabeled tape because nonalphanumeric characters cannot be entered with the operator command.

If a VSN is not associated with the requested lfn, the system directs the operator to assign an available device.

CONTROL STATEMENT RULES

On the tape assignment control statements (LABEL, REQUEST, and ASSIGN), the user can specify the tape label contents, tape density, track type, 9-track conversion mode, data format, noise size, and processing options. If any of these specifications are omitted, the system uses a default value.

NOTE

For 9-track tapes, the density specification given on the tape assignment is used only when the tape is written from load point. Otherwise, the tape is read or written using the density previously used for that tape. To ensure that a labeled tape is at load point for rewriting the tape at a new density, perform one of the following before the write operation.

- Rewind the tape.
- Specify the W parameter on the LABEL statement used to assign the tape.
- Assign the tape using a REQUEST or ASSIGN control statement.

Specification of duplicate or equivalent parameters is not allowed on tape assignment control statements.

NOTE

The user is advised not to create labeled S or L format tapes with tape marks embedded in the data. Future adherence to ANSI standards will make these tapes nonstandard as the ANSI standard allows tape marks to be used only as delimiters of label groups.

The system allows use of a continuation line for an ASSIGN, BLANK, LABEL, REQUEST, and VSN control statement when any one of these requires more than 80 characters. If, in processing one of these statements, the system does not encounter a termination character prior to the end of the line, it assumes the next line is a continuation line. A continuation line should be terminated with a valid terminator. The terminator for a continuation line must appear in or before column 80.

NOTE

The system accepts continuation lines from a time-sharing terminal only if they are within a procedure file.

The programmer can use literals for parameters that contain nonalphanumeric characters. These parameters are FI/L, FA, SI/M, VA, FA, OFA, and VSN. Nonalphanumeric characters are characters other than letters, numbers, and asterisks.

A literal is a character string delimited by dollar signs. Blanks within literals are retained. If the literal is to contain a dollar sign, two consecutive dollar signs must be included. Thus, the literal

```
$A B$$41$
```

is interpreted as:

```
A B$41
```

When continuation lines are used, a literal cannot extend from one line to another.

PROCESSING OPTIONS

The PO= parameter on the LABEL, ASSIGN, and REQUEST tape assignment statements allows the user to specify one or more processing options that are to apply to that tape file. The characters representing the processing options and their meanings are listed below. PO=S gives the default end-of-tape conditions. Default error recovery attempts to recover blocks having errors by repeatedly rereading the block.

p_i	Meaning
A	Job is automatically aborted on an irrecoverable read or write parity error (refer to the N option).
E	Error inhibit. All hardware read/write errors are ignored and processing continues. The system does not attempt error recovery, issue error messages, or return error status. During a read operation, blocks less than noise size (refer to the NS parameter) are unconditionally bypassed. This option is not intended for the normal user. It can be used to recover portions of data from a bad tape, to check out hardware, and to write on tape without skipping bad spots; in the latter case, the user is responsible for verifying that the data is written correctly.
F	Force unload. Unload at end of usage. (Refer to the U option.)
G	Disables all hardware error correction activity in GE (6250 cpi) write mode. An on-the-fly error while writing a GE tape results in standard error recovery processing. The system erases the defective portion of tape, thereby reducing the amount of data that can be stored on the tape. The default is installation-defined (refer to the H option).
H	Enables hardware error correction activity in GE (6250 cpi) write mode. The system allows certain types of single-track errors to be written that can be corrected when the tape is read (on-the-fly correction). This is the recommended mode because it provides efficient throughput, error

P_i	Meaning
	recovery, and tape usage when writing GE tapes on a medium that is suitable for use at 3200 fci or 6250 cpi. The default option (G or H) is installation-defined.
I	If, during a write operation, the system senses the end-of-tape (EOT), it rewrites the block on which the EOT occurred as the first block on the next volume. During a read operation, the block on which the EOT occurred is ignored and reading continues on the next volume. If a tape mark and the EOT are sensed at the same time, the EOT is ignored. This option cannot be specified for I or SI format tapes. Refer to the P and S options.
L	Issues only the first and last error messages for each bad tape block. Numerous attempts are made to read each bad block, but only the messages for the first and last attempts are issued to the dayfile. The default is installation-defined (refer to the M option).
M	Issues an error message for each attempt to read a bad tape block. The default is installation-defined (refer to the L option).
N	Job is not automatically aborted on an irrecoverable read or write parity error (refer to the A option); data is passed to the job on a read operation.
P	If, during a write operation, the system senses the end-of-tape, the system writes a trailer sequence following the block on which the EOT was sensed. Any data that occurs following the block on which EOT was sensed, yet before the tape mark, is ignored. During a read operation, the system transfers the block on which the EOT was sensed to the user job. The read operation resumes on the next reel. If a tape mark and the EOT are sensed at the same time, the EOT is ignored. Refer to the I and S options.
R	Enforce ring out. If the tape is mounted with the write ring in, job processing is suspended until the operator remounts the tape correctly.
S	Specifies where the system is to stop on an exit condition. For unlabeled tape, it directs the system to stop at the first tape mark after the EOT is sensed. For labeled tape, it directs the system to stop at the tape mark plus EOF1 or the tape mark plus EOV1 when the EOT is encountered. If, during a write operation, the system senses the end-of-tape, the system writes a trailer sequence following the block on which the EOT was sensed. This trailer sequence consists of a tape mark followed by an EOV1 label for labeled tapes and four tape marks for unlabeled tapes. The next block is written on the next volume. During a read operation, the EOT is noted and the system transfers to the user job the block on which the EOT was sensed plus all following blocks until a trailer sequence (as described previously) is recognized. Reading resumes on the next volume.

p_i	Meaning
U	Inhibit unload. Do not unload at the end of usage. For system origin jobs, the inhibit unload option is selected by default; for all other jobs, omission of the U option causes the tape to be unloaded at end of usage.
W	Enforce ring in. If the tape is mounted without the write ring in, job processing is suspended until the operator remounts the tape correctly.

If both ring enforcement options (R and W) are specified or more than one EOT option (I, P, or S) is specified, the system issues a dayfile message and terminates the job step.

For further information on end-of-tape/end-of-reel conditions, refer to the CLOSER, REWIND, and UNLOAD macros in section 3 and the LABEL macro in section 4 of volume 2.

ASSIGN STATEMENT

The ASSIGN control statement names a tape unit and the local file to be assigned to that unit. It can create an unlabeled tape file or access an existing labeled or unlabeled tape. It cannot create or verify tape labels.

NOTE

Only system origin jobs or users validated for system origin privileges (DEBUG mode) and for use of magnetic tapes can use the ASSIGN statement to assign a tape unit.

Jobs that use this statement without proper validation are aborted, and a dayfile message is issued.

Before performing the assignment, the system unloads the local file (refer to the UNLOAD statement in section 7).

The following description applies only to magnetic tape files; for use of the ASSIGN statement with devices other than magnetic tape, refer to section 7.

The control statement format is:

$$\text{ASSIGN}(nn, lfn, \text{VSN}=vsn, \left\{\begin{matrix}\text{MT}\\ \text{NT}\end{matrix}\right\}, \left\{\begin{matrix}\text{D}=den\\ den\end{matrix}\right\}, \text{F}=format, \text{LB}=\ell, \left\{\begin{matrix}\text{FC}=fcount\\ \text{C}=ccount\end{matrix}\right\}, \text{CV}=conv,$$
$$\text{NS}=ns, \text{PO}=p_1p_2\ldots p_n, \left\{\begin{matrix}\text{CK}\\ \text{CB}\end{matrix}\right\})$$

Required parameters:

nn — Device or device type to which the file lfn is assigned. nn can be the EST ordinal† of a magnetic tape unit or one of the device types MT or NT. Specifying MT informs the operator to assign the file to a 7-track magnetic tape drive; NT informs the operator to assign the file to a 9-track magnetic tape drive. Omission of this parameter results in an error.

† Contact installation personnel for a list of EST ordinals.

lfn — Name of the file to be assigned to the device nn. Omission of this parameter results in an error.

Optional parameters:

VSN=vsn — A 1- to 6-character volume serial number that uniquely identifies a reel of tape. ASSIGN does not use the VSN parameter to assign the tape. The nn parameter determines the tape assignment.

MT or NT — Specifies 7-track (MT) or 9-track (NT) tape drive. It must not conflict with the nn specification.

D=den or den — Tape density; it must not conflict with the MT or NT specification. Installation-defined default. Ignored for 9-track tapes not positioned at load point. May be one of the following:

7-track (MT)		9-track (NT)	
den	Density	den	Density
LO	200 bpi	HD	800 cpi
HI	556 bpi	PE	1600 cpi
HY	800 bpi	GE	6250 cpi
200	200 bpi	800	800 cpi
556	556 bpi	1600	1600 cpi
800	800 bpi	6250	6250 cpi

F=format — Data format. Default is I. Refer to Magnetic Tape Files in section 2 for descriptions of the data formats.

- I Internal
- SI System internal†
- S Stranger
- L Long block stranger
- F Foreign

LB=*l* — Labeled or unlabeled tape. Default is KU if VSN is omitted or KL if VSN is specified.

- KU Unlabeled.
- KL ANSI-labeled. If the tape is a NOS tape, volume and header label access restrictions are enforced (refer to appendix G).
- NS Nonstandard-labeled. Assumes data begins immediately after the first tape mark.

FC=fcount or C=ccount — Whenever F format is specified, this parameter specifies maximum block size in frames (fcount) or in 6-bit characters (ccount). Ignored for other tape formats. No default value.

†NOS/BE system default tape format (binary mode only); used for tape interchange with NOS/BE systems.

CV=conv — Conversion mode† for 9-track tapes; applies to both labels and data on coded tapes; applies only to labels on binary tapes. Installation-defined default. Ignored for unlabeled I or SI format binary tapes whose trailer labels are always ASCII. Must not be specified with MT or 7-track density specification.

- AS ASCII/display code conversion.
- US Same as AS.
- EB EBCDIC/display code conversion.

NS=ns — Noise size. Ignored for I and SI format tapes. Default is 18 frames for other formats. Maximum value is 31 frames. If NS=0 is specified, the default is used.

PO=$p_1p_2...p_n$ — A string of characters (not separated by commas) that specify processing options (refer to Processing Options in this section).

CK or CB — lfn is to be used as a checkpoint file (refer to section 11).

- CK Each dump is written at the previous EOI of lfn.
- CB Each dump is written at the BOI of lfn.

Example:

```
ASSIGN(51,TAPE1,D=PE,F=SI)
```

This statement assigns the file TAPE1 to the 9-track magnetic tape unit identified by EST ordinal 51.

BLANK STATEMENT

The BLANK control statement writes the ANSI standard labels VOL1, HDR1, and EOF1 following the load point of a tape. The labels are written as follows (asterisks represent tape marks):

	VOL1	HDR1	*	*	EOF1	*	*		

If the value of a labeled field is specified by a BLANK statement parameter, that value is written; otherwise, the default value is used. Refer to appendix G for the tape label formats and default values.

NOTE

A BLANK statement issued in a nonsystem origin job cannot overwrite a label containing an unexpired expiration date or a nonblank VA field.

If the FA field within the label is nonblank, a nonsystem origin job must specify the FA character using the OFA parameter. If the FA character is A, only the owner or a system origin job can overwrite the label.

† Refer to Magnetic Tape Users in appendix A.

The control statement format is:

BLANK(VSN=vsn, {MT / NT}, {D=den / den}, CV=conv, FA=fa, OFA=ofa, VA=va,
OWNER=usernum/familyname, LSL=lsl, U)

VSN=vsn — 1- to 6-character volume serial number that uniquely identifies the reel of tape. It is entered in the VOL1 label. It need not match the VSN previously recorded on the tape.

MT or NT — Specifies 7-track (MT) or 9-track (NT) tape drive. Installation-defined default. Must not conflict with D=den specification.

D=den or den — Tape density; it must not conflict with the MT or NT specification. Installation-defined default. May be one of the following:

7-track (MT)		9-track (NT)	
LO	200 bpi	HD	800 cpi
HI	556 bpi	PE	1600 cpi
HY	800 bpi	GE	6250 cpi
200	200 bpi	800	800 cpi
556	556 bpi	1600	1600 cpi
800	800 bpi	6250	6250 cpi

CV=conv — Conversion mode† for 9-track tape labels. Installation-defined default. Must not be specified with MT or 7-track density specification.

- AS — ASCII/display code conversion.
- US — Same as AS.
- EB — EBCDIC/display code conversion.

FA=fa — File accessibility character indicating who has access to the labeled tape. Value entered in HDR1 and EOF1 labels.

- Blank — Unlimited access (default).
- A — Only the owner of this NOS written tape can access it.
- Other — In all future accesses of this tape, the user must specify this character.

OFA=ofa — Old file accessibility character on a labeled tape that is to be relabeled. This parameter must be specified if the FA field is currently other than A or blank. Future accesses of the tape must specify the character specified with the FA parameter.

VA=va — Volume accessibility character indicating that the valume must be accessed as an ANSI-labeled tape (LB=KL). If VA is nonblank, only a system origin job can change VOL1. Default is unrestricted access. Refer to the VOL1 format in appendix G.

†Refer to Magnetic Tape Users in appendix A.

OWNER=usernum/familyname	Owner identification entered in VOL1 label. Determines the owner for file accessibility (FA) parameter.
LSL=lsl	Label standard level entered in VOL1 label. Default is 1.
1	Tape labels and data format for this volume conform to the ANSI standard.
Blank	Tape labels and data format for this volume may or may not conform to the ANSI standard.
U	If U is specified, the tape is physically unloaded when returned after blank labeling. If U is omitted, physical unloading is inhibited. This parameter does not apply to system origin jobs.

An installation can use the BLANK statement to restrict use of its labeled tapes. Once a tape has been blank labeled, the user can modify the labels as follows:

1. If the volume accessibility field of VOL1 indicates unlimited access (that is, VA is blank), the user can:
   - Include another BLANK statement to change VOL1, HDR1, or EOF1 values.
   - Request the tape as unlabeled (with the parameter LB=KU) and write it in whatever format the user specifies.
   - Include a LABEL statement to change HDR1 by specifying one or more of the parameters associated with that label and specifying the W parameter.
2. If the volume accessibility field is nonblank, the user can:
   - Include a LABEL statement to change HDR1. However, in requesting a tape in which VA is nonblank, the user must specify an ANSI labeled tape (with the parameter LB=KL), and therefore, cannot change or destroy the VOL1 label.
   - If validated, submit a system origin job to change VOL1.

LABEL STATEMENT

Like the ASSIGN and REQUEST statements, the LABEL control statement associates a file name lfn with a magnetic tape, usually identified by its VSN. Unlike the ASSIGN and REQUEST statements, the LABEL statement can create and verify tape labels. It can also position a multifile set for access to any of its existing files or for appending a new file. The LABEL statement can create and access unlabeled as well as labeled tapes.

NOTE

A LABEL statement cannot overwrite a label with an unexpired expiration date (refer to appendix G).

To write the labels that begin a labeled tape (refer to Magnetic Tape Files in section 2), the user should specify a write label (W) parameter. The W parameter always rewinds the tape to load point and rewrites the first label group. The label contents remain the same when a LABEL statement with the W parameter names an lfn already assigned to a tape file.

If the tape was not previously part of a multifile set (the SI field in the first HDR1 label is blank), then specification of the SI and QN=9999 parameters rewrites the initial tape labels.

To position the tape after any HDR1 label other than the first HDR1 label (multifile set only), the SI parameter must be specified. When SI is specified, the R and W parameters are ignored unless QN=1 and the first file on the tape is to be written. The system determines where to position the tape by matching the SI, FI, and QN parameter values (if specified) to the corresponding values in the HDR1 label. (The HDR1 label format is given in appendix G).

To write the EOF1 and HDR1 labels between two files in a multifile set (refer to figure 1-2-2), the user specifies the SI and QN=9999 parameters. The W parameter is ignored if specified when appending a file (QN=9999).

If neither the MT nor NT parameter is specified and no VSN is named, the operator can assign the file to any equipment. The user must be validated for the assigned equipment or the job is terminated.

The control statement format is:

$$\begin{aligned}&\text{LABEL(lfn, VSN=vsn, } \left\{\begin{matrix}\text{MT}\\ \text{NT}\end{matrix}\right\}\text{, D=den, F=format, LB=}\ell\text{, } \left\{\begin{matrix}\text{FC=fcount}\\ \text{C=ccount}\end{matrix}\right\}\text{, CV=conv, NS=ns,}\\ &\quad \text{PO=}p_1p_2\ldots p_n\text{, } \left\{\begin{matrix}\text{CK}\\ \text{CB}\end{matrix}\right\}\text{, } \left\{\begin{matrix}\text{SI=setid}\\ \text{M=setid}\end{matrix}\right\}\text{, } \left\{\begin{matrix}\text{SN=secno}\\ \text{V=secno}\end{matrix}\right\}\text{, } \left\{\begin{matrix}\text{QN=seqno}\\ \text{P=seqno}\end{matrix}\right\}\text{, } \left\{\begin{matrix}\text{FI=fileid}\\ \text{L=fileid}\end{matrix}\right\}\text{,}\\ &\quad \text{FA=fa, G=genno, E=gvn, } \left\{\begin{matrix}\text{CR=cdate}\\ \text{C=cdate}\end{matrix}\right\}\text{, } \left\{\begin{matrix}\text{RT=yyddd}\\ \text{T=ddd}\end{matrix}\right\}\text{, } \left\{\begin{matrix}\text{W}\\ \text{R}\end{matrix}\right\}\text{)}\end{aligned}$$

Required parameter:

lfn	Name of the file that resides or is to reside on magnetic tape. If lfn is already assigned to a mass storage file, processing continues with the next control statement. To assign a previously assigned lfn, the user must return lfn before its reassignment. If lfn is already assigned to a tape and the R parameter is specified, the contents of the tape labels are compared to the statement parameter specifications. If the label verification fails, the job aborts.

Optional parameters:

VSN=vsn	A 1- to 6-character volume serial number that uniquely identifies a reel of tape. If VSN is omitted, the operator assigns an available unit to lfn. Multiple VSNs can be specified if separated by / or = characters. If the VSNs are separated by the = character, LABEL assigns lfn to the first available VSN in the list. If the VSNs are separated by the / character, lfn is a multivolume file set, and LABEL assigns the volumes in the sequence given.
MT or NT	Requests 7-track (MT) or 9-track (NT) tape drive. Installation-defined default. Must not conflict with D=den specification.
D=den	Tape density; must not conflict with MT or NT specification. Installation-defined default. Ignored for 9-track tape not positioned at load point. Can be one of the following.

7-track (MT).		9-track (NT)	
den	Density	den	Density
LO	200 bpi	HD	800 cpi
HI	556 bpi	PE	1600 cpi
HY	800 bpi	GE	6250 cpi
200	200 bpi	800	800 cpi
556	556 bpi	1600	1600 cpi
800	800 bpi	6250	6250 cpi

F=format — Data format. Default is I. Refer to Magnetic Tape Files in section 2.

- I — Internal
- SI — System internal†
- S — Stranger
- L — Long block stranger
- F — Foreign

LB=*l* — Labeled or unlabeled tape. Default is KL.

- KL — ANSI-labeled.
- KU — Unlabeled.
- NS — Nonstandard-labeled. Assumes data begins immediately after the first tape mark.

FC=fcount or C=ccount — Whenever F format is specified, this parameter must specify the maximum block size in frames (no default value). Ignored for other tape formats.

CV=conv — Conversion mode†† for 9-track tapes; applies to both labels and data on coded tapes; applies only to labels on binary tapes. Installation-defined default. Ignored for unlabeled I or SI format binary tapes whose trailer labels are always ASCII. Must not be specified with MT or 7-track density specification.

- AS — ASCII/display code conversion.
- US — Same as AS.
- EB — EBCDIC/display code conversion.

NS=ns — Noise size; any block containing fewer than ns frames is considered noise and discarded. Ignored for I and SI format tapes. Default is 18 frames for other formats. Maximum value is 31 frames. If NS=0 is specified, the default is used.

PO=$p_1p_2...p_n$ — A string of characters (not separated by commas) that specifies processing options. Refer to Processing Options in this section.

†NOS/BE system default tape format (binary mode only); used for tape interchange with NOS/BE systems.

††Refer to Magnetic Tape Users in appendix A.

CK
or
CB

lfn is to be used as a checkpoint file (refer to section 11).

- CK Each dump is written at the previous EOI of lfn.
- CB Each dump is written at the BOI of lfn.

Optional tape label parameters (refer to appendix G):

SI=setid

1- to 6-character file set identifier; must be specified for file positioning within a multifile set. The default is blank.

NOTE

To conform to the ANSI standard for tape interchange when creating a multifile set, the user must specify the same setid for all files within the set.

SN=secno

One-to four-digit file section number specifying the position of the volume within a multivolume file set (numbered consecutively from 0001). The default is 1.

QN=seqno

One-to four-digit file sequence number specifying the position of the file within the multifile set (numbered consecutively from 0001). The default is 1. QN must be set to 9999 to append a new file to a multifile set.

FI=fileid
or
L=fileid

1- to 17-character file identifier recorded in the HDR1 label (refer to appendix G). The default is blank.

FA=fa

File accessibility character indicating who has access to the labeled tape.

- Blank Unlimited access (default).
- A Only the owner of the tape can access it.
- Other To access the tape, the user must specify the character in the FA field of the HDR1 label.

G=genno

One-to four-digit generation number. The default is 1.

E=gvn

One-to two-digit generation version number. The default is 0.

CR=cdate

Creation date in the form yyddd where $1 \leq ddd \leq 366$. Used only on read operations; write operations always use the current date.

RT=yyddd
or
T=ddd

Used to determine the expiration date entered in the HDR1 label. On or after this date the label and the file can be overwritten. R=yyddd specifies the expiration date where yy is the last two digits of the year, and ddd is the day of the year ($1 \leq ddd \leq 366$). T=ddd specifies the number of days the file is to be retained ($1 \leq ddd \leq 999$).

R
or
W

If R is specified, the system compares the values recorded on the file labels with the LABEL statement parameter values. If the comparison fails, it terminates the job. R is the default.

If W is specified, the system writes ANSI standard labels containing the values specified with the LABEL statement parameters or their default values. If the tape is mounted without the write ring, job processing is suspended until the operator remounts the tape correctly. If both the W and the PO=R parameters are specified, the job step aborts.

W and R are ignored when SI is specified and QN>1. When QN=1 (default value) and W are specified, the initial header is rewritten..

Example 1:

```
LABEL(NEWFILE,VSN=TP01,FI=FILEA,W)
```

This statement creates an ANSI-labeled tape which the job can access by the file name NEWFILE. Default values are used for all fields of HDR1 except the file identification, FILEA. Any data written is recorded in the default I format.

Example 2:

```
LABEL(OLDFILE,VSN=TP01,FI=FILEA)
```

This statement assigns the tape file created in a previous job (refer to example 1) to the file name OLDFILE. The system compares the VSN in the VOL1 label and the file identification in the HDR1 label with the values on the statement.

Example 3:

The following sequence of control statements in a single job creates two files of a multifile set.

```
LABEL(TAPE,VSN=ONE,D=PE,F=I,FI=FIRSTFILE,SI=TEST,QN=1,W)
COPYBR(INPUT,TAPE)
LABEL(TAPE,FI=SECONDFILE,SI=TEST,QN=9999)
COPYBR(INPUT,TAPE,10)
RETURN(TAPE)
```

The first statement writes the initial labels on the tape for the first file in the file set. The second label statement specifies QN=9999 to add a second label group. Future references to the second file of the file set must specify QN=2 (refer to examples 6, 7, and 8).

Example 4:

The following control statements in another job add the third label group and file to the file set created in example 3.

```
LABEL(TAPE,VSN=ONE,D=PE,F=I,FI=THIRDFILE,SI=TEST,QN=9999)
COPYBR(DISK,TAPE,3)
RETURN(TAPE)
```

Example 5

Any one of the following control statements can be used to read the first file of the file set created in examples 3 and 4.

```
LABEL(TAPE, VSN=ONE, D=PE, F=I)
LABEL(TAPE, VSN=ONE, D=PE, F=I, FI=FIRSTFILE)
LABEL(TAPE, VSN=ONE, D=PE, F=I, FI=FIRSTFILE, SI=TEST)
```

The last two statements position the tape according to the file identifier specified. This method is used if the sequential location of the file on the tape is not known.

```
LABEL(TAPE, VSN=ONE, D=PE, F=I, QN=1, SI=TEST)
```

This statement positions the tape according to the sequence number of the file.

```
LABEL(TAPE, VSN=ONE, D=PE, F=I, QN=1, FI=FIRSTFILE, SI=TEST)
```

This statement positions the tape according to the sequence number specified, but the file identifier specified must also be correct or the job aborts.

Example 6

Any one of the following control statements can be used to read the second file of the multifile set created in example 3.

```
LABEL(TAPE, VSN=ONE, D=PE, F=I, QN=2, SI=TEST)
LABEL(TAPE, VSN=ONE, D=PE, F=I, FI=SECONDFILE, SI=TEST)
LABEL(TAPE, VSN=ONE, D=PE, F=I, FI=SECONDFILE, QN=2, SI=TEST)
```

Example 7

The following control statements destroy the third file of the multifile set created in example 3.

```
LABEL(TAPE, VSN=ONE, D=PE, F=I, QN=2, SI=TEST)
COPYBR(DISK, TAPE)
REWIND(TAPE)
```

The tape is positioned at the beginning of file 2, a new file 2 is written, and the tape is rewound, thereby writing a trailer sequence after file 2.

Example 8

The following example replaces the second file of the multifile set created in example 3 and retains the first and third files.

```
LABEL(TAPE, VSN=ONE, D=PE, F=I, QN=3, SI=TEST)
COPYBR(TAPE, DISK, 3)
LABEL(TAPE, VSN=ONE, D=PE, F=I, QN=2, SI=TEST)
COPYBR(INPUT, TAPE)
LABEL(TAPE, VSN=ONE, D=PE, F=I, QN=9999, SI=TEST, FI=THIRDFILE)
REWIND(DISK)
COPYBR(DISK, TAPE, 3)
```

The first copy saves file 3, the second copy replaces file 2, and the third copy copies back file 3. The third LABEL statement rebuilds the labels for file 3.

LISTLB STATEMENT

The LISTLB control statement lists the labels of an ANSI-labeled tape file previously assigned the file name lfn.

The control statement format is:

LISTLB(lfn, {SI=setid / M=setid}, {QN=seqno / P=seqno}, LO=ltype, L=out)

lfn — File name assigned to tape file whose labels are to be listed.

SI=setid or M=setid — 1- to 6-character file set identifier. If specified, only label groups whose HDR1 label contains this value are listed.

QN=seqno — One-to four-digit file sequence number. If seqno is specified, only the label group whose HDR1 label contains this value is listed. If seqno is specified, SI must be specified; otherwise, LISTLB terminates.

LO=ltype — Label type(s) to be listed. The default is R. Required and optional labels are listed in appendix G. Combinations of ltype mnemonics can be specified, such as LO=VH to list only the VOLn and HDRn labels.

A	List all labels.
R	List required labels.
O	List optional labels.
V	List VOLn labels.
H	List HDRn labels.
F	List EOFn labels.
E	List EOVn labels.
U	List UVLn, UHLn, and UTLn labels.

L=out — File on which the labels are to be listed. Default is OUTPUT.

To list labels for a multifile set (lfn contains more than one HDR1/EOF1 label pair), the tape must be positioned at load point. LISTLB then positions the tape for reading the requested labels. It searches for labels through all volumes associated with lfn. At the end of the multifile set or if an expected label group is not found, the following dayfile message is issued. n is the sequence number of the last file found. (nnn should be ignored.)

```
MULTI-FILE NOT FOUND, lfn AT nnn.
REQUEST SECTION n+1.
FOUND SECTION n.
```

After issuing this dayfile message, LISTLB leaves the tape positioned after the last listed label. The next statement processed for the tape file must be either RETURN, EVICT, UNLOAD, or LABEL.

Example 1:

The following statements list the second label group of file set ABCDEF.

```
LABEL(T,VSN=EXAMP1,MT,D=HY,SI=ABCDEF)
LISTLB(T,SI=ABCDEF,QN=2)
```

Example 2:

To list only the volume and header labels (trailer labels omitted) of a multivolume file set, the user must request a volume of the file set, list its labels, and return the file set, repeating the procedure for each volume of the file set.

```
LABEL(T,VSN=REEL1,MT,D=HY)
LISTLB(T,LO=VH)
RETURN(T)
LABEL(T,MT,D=HY,VSN=REEL2)
LISTLB(T,LO=VH)
```

Example 3:

To list all labels of the following file set, only one LISTLB control statement is required.

```
VSN(T=REEL1/REEL2)
LABEL(T,VSN=REEL1,D=HY)
LISTLB(T)
```

The LISTLB(T) statement lists all labels on the volumes associated with T, that is, REEL1 and REEL2.

REQUEST STATEMENT

The REQUEST control statement associates a file name, lfn, with a magnetic tape device,† usually described in a comment following the statement terminator. This comment is displayed at the system console, directing the operator to make the requested assignment. However, if the tape is labeled and the user previously specified a VSN via the VSN control statement or included the VSN parameter on the REQUEST statement, the system can automatically assign the tape.

NOTE

Users should avoid issuing the REQUEST statement because it prevents use of a control point while the operator responds to the tape request. The LABEL statement is recommended for tape assignment.

† If the user does not specify a VSN parameter or an MT or NT parameter on the statement, the operator can assign any device to the file. If the user is not validated for the assigned device, the job aborts.

The REQUEST statement can create unlabeled tape files and access existing labeled and unlabeled tape files. It cannot create or verify tape labels.

The control statement format is:

REQUEST(lfn, VSN=vsn, {MT / NT}, {D=den / den}, F=format, LB=*l*, {FC=fcount / C=ccount}, CV=conv,
NS=ns, PO=$p_1p_2 \ldots p_n$, {CK / CB})

Required parameter:

lfn	Name of the file that resides or is to reside on magnetic tape. If lfn is already assigned to a mass storage file, processing continues with the next control statement. To assign a previously assigned lfn, the user must return lfn before its reassignment.

Optional parameters:

VSN=vsn	1- to 6-character volume serial number uniquely identifying a reel of tape. The user should specify a VSN for labeled and unlabeled tapes. If VSN is omitted, the operator must assign an available device to lfn.
MT or NT	Requests 7-track (MT) or 9-track (NT) tape drive. Installation-defined default. Must not conflict with D=den specification.
D=den or den	Tape density; must not conflict with MT or NT specification. Installation-defined default. Ignored for 9-track tape not positioned at load point. May be one of the following.

7-track (MT)		9-track (NT)	
den	Density	den	Density
LO	200 bpi	HD	800 cpi
HI	556 bpi	PE	1600 cpi
HY	800 bpi	GE	6250 cpi
200	200 bpi	800	800 cpi
556	556 bpi	1600	1600 cpi
800	800 bpi	6250	6250 cpi

F=format	Data format. The default is I. Refer to Magnetic Tape Files in section 2 for format descriptions.

I	Internal
SI	System internal†
S	Stranger
L	Long block stranger
F	Foreign

LB=*l*	Labeled or unlabeled tape. The default is KL if a volume serial number is specified by the VSN parameter or by a VSN control statement; otherwise, the default is KU.

†NOS/BE system default tape format (binary mode only); used for interchange with NOS/BE systems.

KL ANSI-labeled.
KU Unlabeled.
NS Nonstandard labeled. Assumes that data begins immediately after the first tape mark.

FC=fcount or C=ccount — Whenever F format is specified, this parameter must specify the maximum block size in frames (no default value). Ignored for other tape formats.

CV=conv — Conversion mode† for 9-track tapes; applies to both labels and data on coded tapes; applies only to labels on binary tapes. Installation-defined default. Ignored for unlabeled I or SI format binary tapes whose trailer labels are always ASCII. Must not be specified with MT or 7-track density specification.

AS ASCII/display code conversion.
US Same as AS.
EB EBCDIC/display code conversion.

NS=ns — Noise size. Ignored for I and SI format tapes. Default is 18 frames for other formats. Maximum value is 31 frames. If NS=0 is specified, the default is used.

PO=$p_1p_2...p_n$ — A string of characters (not separated by commas) specifying processing options (refer to Processing Options in this section).

CK or CB — lfn is to be used as a checkpoint file (refer to section 11).

CK Each dump is written at the previous EOI of lfn.
CB Each dump is written at the BOI of lfn.

Example:

To send a message to the operator requesting that volume XYZ be mounted on tape unit NT62 and assigned to lfn TAPE1, the user could issue the following statement.

```
REQUEST,TAPE1. NEED VSN=XYZ ON NT62.
```

VSN STATEMENT

The VSN control statement associates a file name lfn with one or more volumes of tape.†† An lfn/VSN association allows the system to assign the specified VSN to lfn without reference to a VSN parameter on the LABEL or REQUEST statement or to an operator command. Once declared, an lfn/VSN association remains until the file is returned.

The control statement format is:

VSN(lfn_1=vsn_1,lfn_2=vsn_2,...,lfn_n=vsn_n)

†Refer to Magnetic Tape Users in appendix A.
††Up to 55 VSNs can be specified for a single file name in any combination of duplicate reel and/or multireel specifications.

lfn_i File name to be associated with vsn_i.

vsn_i One or more 1- to 6-character volume serial numbers to be associated with lfn_i. If vsn_i contains nonalphanumeric characters, it must be a literal [delimited by dollar signs ($)].

vsn_i	Meaning
omitted	An available scratch tape is automatically assigned to lfn_i.
0	Same as omitted.
SCRATCH	Same as omitted.
$vsn_1=vsn_2=...=vsn_n$	Names duplicate volumes, any of which may be used with lfn_i.
$vsn_1/vsn_2/.../vsn_n$	Successive volumes to be assigned to lfn_i.† The system assigns volumes in the order listed.

With a VSN statement the user can:

- Omit the VSN keyword from his LABEL or REQUEST statements and specify lfn/VSN associations on the VSN statement instead. This allows the user to specify new VSNs without changing LABEL or REQUEST statements.
- Override the VSN specified on subsequent ASSIGN, LABEL, REQUEST, or VSN statements. For example, the sequence

```
VSN(FILEA=123)
VSN(FILEA=124)
LABEL(FILEA)
```

  directs the system to assign FILEA to the tape with VSN 123. However, if the user returns the file lfn, he can specify another lfn/VSN association. Thus, the following sequence directs the system to assign FILEA to the tape with VSN 124.

```
VSN(FILEA=123)
RETURN(FILEA)
VSN(FILEA=124)
LABEL(FILEA)
```

- Associate the VSNs of two or more duplicate volumes with one file name. For example, the following statement indicate that either the tape with VSN VOL100 or the tape with VSN VOL101 can be assigned to FILE1.

```
VSN(FILE1=VOL100=VOL101)
```

†All subsequent volumes must have the same characteristics as the first volume in the sequence. (Characteristics include labels, track type, density, and conversion mode.) It is recommended that all volumes be blank labeled (refer to the BLANK statement) before use in a multivolume sequence.

- Specify the VSNs of a multivolume file set. For example, the following statement indicates that FILE2 may extend through the three volumes identified by VSN23, VSN24, and VSN25.

  VSN(FILE2=VSN23/VSN24/VSN25)

CHECKPOINT/RESTART 11

A job may terminate as the result of system, operator, or programmer error. For some jobs, it becomes more advantageous to accept the overhead of checkpoint procedures than to run the risk of losing the entire job output. The checkpoint/restart feature is implemented through the CKP control statement and the RESTART control statement.

NOTE

For information concerning security restrictions associated with the use of these control statements, refer to Security Control in section 3.

CKP STATEMENT

The CKP control statement causes a checkpoint dump to be taken.

The control statement format is:

CKP($lfn_1, lfn_2, \ldots, lfn_n$)

lfn_i Specifies a file to be included in the checkpoint dump. If no files are specified, all files local to the job at the time the CKP statement is processed are checkpointed.

Each time a CKP statement is processed, the system takes a checkpoint dump. The dump is written on the tape or mass storage checkpoint file specified on a REQUEST, ASSIGN, or LABEL control statement with the CK or CB parameter. The dump consists of a copy of the user's central memory, the system information used for job control, and the names and contents of all assigned files explicitly or implicitly identified by the CKP statement. These files are:

- INPUT, OUTPUT, PUNCH, PUNCHB, P8, CCCCCCO, and LGO. These files are always included in the checkpoint dump. CCL ZZZZZxx working files are also included if present.
- Common files, library type files, working copies of indirect access files, and some direct access files. If one of these types of files is specified on the CKP statement, it is included in the checkpoint dump, and all other files of that type are excluded. If no files are specified, all files of these types assigned to the job are included in the dump.

Each checkpointed file is copied according to the last operation performed on it. If the last operation was a write, the file is copied from the BOI to its position at checkpoint time; only that portion is available at restart time. The file is positioned at the latter point.

If the last operation was a read and the EOI was not detected, the file is copied from its position at checkpoint time to the EOI; only that portion is available at restart time. The file is positioned at the former point. If the last operation was a read and the EOI was detected, no copy is performed.

The exception to this rule is the type of operation performed on execute-only direct access files. If a dump is specified for this type of file, its name and associated system information are copied but the contents of the file itself is not copied. Thus, if the user attempts to resume from such a dump, RESTART is unable to retrieve that file and aborts. The user can avoid this by selecting the NA and FC options of the RESTART statement and retrieving the file himself.

If the checkpoint file is to reside on mass storage, the user must include a SAVE or DEFINE control statement in the checkpoint job and a GET or ATTACH control statement in the restart job.

If the checkpoint file is to reside on magnetic tape, care should be taken to use a labeled or nonblank tape. An unlabeled blank tape (one which has never been used) cannot be specified as the checkpoint file since the checkpoint program attempts to read the tape to determine the number of the last checkpoint. The tape subsystem then aborts the job with a blank tape read message.

The system numbers checkpoints starting at 1 and increments by 1 to a limit of 4095. At this point, a second cycle of numbering begins, again starting at 1. An example showing how to restart from a specific checkpoint is given in the RESTART control statement section.

RESTART STATEMENT

The RESTART control statement directs the system to restart a previously terminated job from a specified checkpoint.

The control statement format is:

RESTART(lfn, nnnn, x_i)

lfn — Identifies the checkpoint file; the user must have write permission to lfn.

nnnn — Number of the checkpoint from which to restart; if nnnn is *, the last available checkpoint on lfn is used; if nnnn is omitted, the first checkpoint is used. The nnnn parameter can be obtained from the CHECKPOINT nnnn COMPLETE messages issued to the user's dayfile in response to CKP control statements.

x_i — Any of the following in any order:

- RI — If this parameter is included, the control statement file on lfn is not restored. The control statement file of this restart job at its current position is used instead. If this parameter is not included, the entire control statement file of the checkpointed job is restored and set to its position at checkpoint time; any control statements following RESTART are not processed.
- NA — If this parameter is included, RESTART does not abort if a required file is not available. Also, if NA is included and a read parity error occurs in an attempt to obtain a file from checkpoint nnnn, RESTART selects checkpoint nnnn-1 if it is available.

FC Normally RESTART restores all files included in the specified checkpoint. However, if this option is selected, RESTART first checks if a file is already local to the restart job. If it is, RESTART does not replace it with the file on the checkpoint dump.

The user must assign lfn to his job before the RESTART statement is processed. He must include a REQUEST, ASSIGN, or LABEL control statement if lfn resides on magnetic tape or a GET or ATTACH control statement if lfn resides on mass storage.

Checkpoint dumps are numbered in ascending order from 1 to 4095. When nnnn equals 4095, the numbering sequence begins again at nnnn equal to 1. The value of nnnn depends on the structure of the checkpoint file, as defined by the CK and CB parameters of the REQUEST, ASSIGN, or LABEL control statements.

If CK was specified when the checkpoints occurred, each dump is appended to the checkpoint file, and therefore, all dumps up to the time the job aborted are available for restart. The user may specify a particular checkpoint dump in the following manner.

Assume a CK file of the name CHKFILE is being used and checkpoint number 4095 has been passed. The job is terminated at checkpoint number 10 in the second cycle of numbering. To restart the job from checkpoint 4 of the second numbering cycle, the following control statements can be used.

SKIPR(CHKFILE,8196)	There are two records for every checkpoint, and 4098 checkpoints must be skipped to reach checkpoint 4 of the second numbering cycle.
COPYBR(CHKFILE,AA,2)	The fourth checkpoint is copied to file AA. At this point, file CHKFILE is not positioned correctly for subsequent checkpoints. If the user intends to continue checkpointing on this file, a BKSP,CHKFILE. statement should be included.
RESTART(AA...)	The job is restarted from file AA using the fourth checkpoint.

If the CB parameter was specified on the ASSIGN, LABEL, or REQUEST statement naming the checkpoint file, each dump is written over the preceding dump, and therefore, only the last dump is available. If two REQUEST, ASSIGN, or LABEL statements with CB specified are submitted, successive dumps are alternated between two files; therefore, the last two dumps are available. †

If the CK parameter is specified for alternate files or if more than two checkpoint files are specified, the system issues a dayfile message and aborts the job.

†If alternate checkpoint files are used and a read parity error occurs in an attempt to read the last checkpoint, RESTART aborts even if the NA option was selected.

All files copied by RESTART are made local to the restart job. Therefore, the user must make sure that any direct access files are not lost. For example, assume that direct access files X, Y, and Z are attached to a job. The job is then checkpointed and X, Y, and Z are copied to the checkpoint file lfn. To retain these files as direct access files during restart, the user should include the following sequence of control cards.

```
PURGE(X, Y, Z)
DEFINE(X, Y, Z)
RESTART(lfn, nnnn, x_i)
```

If the information table associated with a file was included on the checkpoint file, but the file itself was not copied, RESTART issues the appropriate commands to retrieve the file.

DEBUGGING AIDS 12

Program errors include errors that prevent compilation or assembly of a source program and errors that prevent execution of an object program. A programmer determines the causes of compilation errors using the compiler diagnostics, a source listing, and the compiler reference manual. The cause of an execution error is often more difficult to determine. If the programmer cannot determine the cause of the error from the execution error message, he can use the CDC CYBER Interactive Debug Utility or interpret memory dumps to locate the cause. CYBER Interactive Debug is described in its reference manual (listed in the preface). This section describes central memory dumps and their use as a debugging aid.

A programmer can dump the job exchange package and locations within the job field length using control statements described in section 9. The exchange package shows the program location where execution ended and the contents of the CPU registers at that time.

A programmer interprets a memory dump using the variable locations given in the program load map and the display code equivalences given in appendix A.

CENTRAL MEMORY DUMPS

The first line of a dump gives the boundaries of the memory that is dumped, relative to the user's field length. Four central memory words are printed per line, with the address of the leftmost word printed on the left-hand side of the page. When the phrase DUPLICATED LINES appears within the dump, all groups of four words not printed are exactly like the last group of four words. Each word is divided into four groups of 15_{10} bits, with the octal representation printed. Figure 1-12-10 is an example of a central memory dump.

EXCHANGE PACKAGE DUMPS

The user can dump his exchange package using a DMP or DMD statement (refer to section 9). Figures 1-12-1 and 1-12-2 show actual exchange package dumps. The format of the first dump is produced by a CYBER 170 Model 171, 172, 173, 174, 175, 720, 730, 750, or 760; a CYBER 70, Model 71, 72, 73, or 74; or a CDC 6000 Series Computer System. The second dump format is produced only by the CYBER 170 Model 176 Computer System.

```
EXCHANGE PACKAGE.

P       242  A0  51760  B0      0      (A0)  0000 0000 0000 0000 0000
RA   622400  A1      1  B1      1      (A1)  0516 0420 0000 0000 0000
FL    52000  A2    114  B2     30      (A2)  0400 0005 0300 0007 7775
EM     7007  A3    574  B3      6      (A3)  5555 5555 5555 5555 5555
RAE       0  A4    557  B4     22      (A4)  7777 7777 7777 7777 7776
FLE       0  A5    573  B5      1      (A5)  1717 0631 4631 4631 4632
MA     3600  A6      1  B6   7776      (A6)  0516 0420 0000 0000 0000
             A7    277  B7  14657      (A7)  3232 3232 3206 0300 0171

X0   7777 7777 7777 0000 0000
X1   0000 0000 0000 0000 0000
X2   0000 0000 0000 0000 0000
X3   0000 0000 0040 0000 0000
X4   2000 0000 0000 0000 0012
X5   1717 0631 4631 4631 4632
X6   0516 0420 0000 0000 0000
X7   0000 0000 0000 0000 0000

(RA)    0000 0000 0000 0000 0000
(RA+1)  0516 0420 0000 0000 0000
```

Figure 1-12-1. Exchange Package Dump

```
EXCHANGE PACKAGE.

P     10435  A0   2165  B0      0      (A0)  1725 2420 2524 0000 0131
RA   136100  A1      1  B1      1      (A1)  0516 0420 0000 0000 0000
FL    15000  A2   6251  B2 777755      (A2)  1717 0631 4631 4640 3615
PSD   70000  A3      2  B3   6032      (A3)  0000 0000 0000 0000 0000
RAE       0  A4   6207  B4  11437      (A4)  0400 0062 4600 0000 0000
FLE       0  A5   4324  B5  12711      (A5)  2000 0000 0000 0000 0065
MA     1200  A6      1  B6 776677      (A6)  0516 0420 0000 0000 0000
EEA    1200  A7  12557  B7     30      (A7)  6000 0000 0000 0001 5000

X0   0000 0000 0000 0000 0000
X1   0000 0000 0000 0000 0000
X2   1717 0631 4631 4640 3615
X3   2000 0000 0000 0000 0012
X4   2000 0000 0000 0000 0000
X5   0000 0000 0000 0000 0000
X6   0516 0420 0000 0000 0000
X7   2000 0000 0000 0000 0001

(RA)    0000 0000 0000 0000 0000
(RA+1)  0516 0420 0000 0000 0000
```

Figure 1-12-2. Exchange Package Dump for CYBER 170 Model 176

The following are the exchange package fields and their contents.

Label	Contents
P	Program address at which execution stopped.
RA	Reference address; starting address of central memory field length.
FL	Field length in central memory.
EM†	Exit mode. Each bit set indicates that if this hardware-detected error occurs, the program aborts. The bit positions are numbered with 0 as the rightmost bit.

Bit Position	Error
11	CM data error. †† eration parity error. ††
10	Central memory control (CMC) input error. ††
9	ECS flag register operation parity error. ††
5-8	Not used.
3-4	Hardware error exit status bits. †††
2	Indefinite operand.
1	Operand out of range.
0	Address out of range.

The EM field in figure 1-12-1 has bit positions 11, 10, 9, 2, 1, and 0 set.

PSD†††† Program status designator (PSD) register. Each bit set indicates the setting of a mode flag or an error condition. The bit positions are numbered with 0 as the rightmost bit.

Bit Positions	Error
17	Exit mode.
16	Monitor mode.
15	Step mode.
14	Indefinite mode.
13	Overflow mode.
12	Underflow mode.

†Does not apply to CYBER 170 model 176.
††Applies to CYBER 170 models 171, 172, 173, 174, 175, 720, 730, 750, and 760 only.
†††Applies to CYBER 70 model 74 only.
††††Applies to CYBER 170 model 176 only.

Label	Contents	
	Bit Position	Error Conditions
	11	LCME error.
	10	CM error.
	9	LCME block range error.
	8	CM block range error.
	7	LCME direct range error.
	6	CM direct range error.
	5	Program range error.
	4	Not used.
	3	Step condition.
	2	Indefinite condition.
	1	Overflow condition.
	0	Underflow condition.

The PSD field in figure 1-12-2 has bit positions 14, 13, and 12 set.

RAE	ECS reference address; starting address of ECS field length.
FLE	ECS field length.
MA	Monitor address (normal exit address for the CYBER 170 model 176).
EEA	Error exit address (CYBER 170 model 176).
Ai	Contents of the address registers.
(Ai)	Contents of the central memory word addressed by the named address register.
Bi	Contents of the increment registers.
Xi	Contents of the operand registers.
(RA)	Contents of the reference address word.
(RA+1)	Contents of the request word following the reference address word.

GENERATING MEANINGFUL DUMPS

The following methods are used to generate meaningful central memory dumps.

- Error Exit Control

  By using the EREXIT macro within his COMPASS program, the user can direct execution when certain errors occur, rather than having his program completely halt execution. This enables him to use it as a checkpoint method (that is, to save generated data to this point). It could also enable him to do further calculations or to write pertinent data to an output file. Refer to section 6, volume 2 for a description of this macro.

- EXIT/NOEXIT/ONEXIT Control

  Once program execution ceases, due to an error condition, and control statement processing is resumed, the user can direct which statements are to be processed through the use of the EXIT, NOEXIT, and ONEXIT statements. Upon an error condition, the user can issue the DMP control statement to obtain appropriate dumps. For a detailed description of these control statements, refer to section 6.

- Dumps may also be generated under control of the user's program through the use of the SYSTEM macro. The FORTRAN user can generate dumps by calling the DUMP subroutine.

READING CM DUMPS

Figures 1-12-3 through 1-12-11 are output from a FORTRAN program source deck processed by the following sequence of control statements.

```
DREW.
USER(EFD252,D)
CHARGE(599,PASS3)
SETCORE,0.
MAP(ON)
FTN,EL=F.
LGO.
OVL.
DMP.
DMP,4500.
```

The source deck in the example consists of four parts.

- Main program (main overlay)
- Function subprogram
- Subroutine subprogram
- Primary overlay

Each part is listed separately followed by the corresponding address assignments, such as variable assignments, program length, and common blocks (refer to figure 1-12-3).

Figures 1-12-7 and 1-12-8 illustrate the load map generated by the MAP control statements. The load map gives the address and references of all entry points. Maps are listed separately for each overlay. Output generated by the program follows the load map (refer to figure 1-12-9).

Figures 1-12-10 and 1-12-11 illustrate central memory dumps generated by the DMP. and DMP, 1000. control statements, respectively.

The following examples illustrate the use of these dumps to obtain specific information.

Example 1: (Finding Data Locations in a Core Dump)

Referring to figure 1-12-3, the variable I is used as the control variable in the DO loop defined by statements 16 through 20. To find the value of I at job termination, the following steps must be performed.

1. Find I in the variable assignments (lower half of figure 1-12-3), noting that I is at relative address 4217_8.
2. Find the first word address (FWA) of the main overlay TESTA. (Refer to the load map, figure 1-12-7.) The FWA of TESTA is 153_8.
3. Add ($153_8 + 4217_8 = 4372_8$) to obtain the absolute address of I.
4. In figure 1-12-11, address 4372 contains 0013_8 (11_{10}). This is the last value of I.

Example 2: (Finding Data Locations in a Core Dump)

To find the variable B(3), the following points must be considered.

- Find B in the variable assignments (lower half of figure 1-12-3). The value is 12, which means that B begins at relative location 12_8 of common block AAA. By referring to the map (figure 1-12-7), note that AAA begins at absolute address 111_8. Therefore, $111_8 + 12_8$ (relative location of B) equals 113_8, the beginning address of array B. B(1) is 123_8, and the address of B(3) is 125_8.
- The location in core of the B array is illustrated in figure 1-12-11.

Example 3: (Finding an Address Within the Program)

Referring to figure 1-12-10, note that the program stopped at address 10004 (the value of P). To find where this is in the program, the following points must be considered.

- Figure 1-12-7 or 1-12-8 contains the routine addresses.
- Figure 1-12-7 illustrates that routine CPUSYS is at address 10001. This means the program ended in routine CPUSYS.

```
      1              OVERLAY(OVL,0,0)
                     PROGRAM TESTA(INPUT,OUTPUT)
                     COMMON//E(100)
                     COMMON/AAA/A(10),B(10),C(3,3)
      5              COMMON/BLOCKA/BLK(5)
                     DIMENSION N(50)
                     DATA (A(I),I=1,10)/1.,2.,3.,4.,5.,6.,7.,8.,9.,10./
                     DATA (B(I),I=1,5)/100.,200.,300.,400.,500./
                     DATA (B(I),I=6,10)/600.,700.,800.,900.,1000./
     10              DATA C(1,1),C(1,2),C(1,3)/101.,202.,303./
                     DATA C(2,1),C(2,2),C(2,3)/2.1,2.2,2.3/
                     DATA C(3,1),C(3,2),C(3,3)/3.1,3.2,3.3/
                     DATA (N(J),J=1,50)/50*123/
                     CALL OVERLAY(4HOVLA,1,0)
     15              CALL PRNT(BLOCKA)
                     DO 30 I=1,10
                     A(I)=A(I)*A(I)
                     A(I)=TRY(A(I),A(I))
                  30 B(I)=TRI(A(I),A(I))
     20              DO 35 J=1,5
                  35 BLK(J) = A(J)+A(2*J)
                     CALL OVERLAY (4HOVLA,1,0)
                     CALL PRNT(BLOCKA)
                     C(1,1)=J
     25              END


        SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
 4137  TESTA

VARIABLES     SN  TYPE          RELOCATION
    0  A          REAL       ARRAY    AAA          12  B          REAL       ARRAY    AAA
    0  BLK        REAL       ARRAY    BLOCKA     4216  BLOCKA     REAL
   24  C          REAL       ARRAY    AAA           0  E          REAL       ARRAY    / /
 4217  I          INTEGER                        4220  J          INTEGER
 4221  N          INTEGER    ARRAY

FILE NAMES       MODE
    0  INPUT                  2054  OUTPUT

EXTERNALS          TYPE   ARGS
       OVERLAY                3                        PRNT               1
       TRI        REAL        2                        TRY        REAL    2

STATEMENT LABELS
    0  30                                  0  35

LOOPS  LABEL    INDEX     FROM-TO    LENGTH    PROPERTIES
 4145  30     * I           16 19       14B             EXT REFS
 4164  35       J           20 21        4B     INSTACK

COMMON BLOCKS   LENGTH
       / /          100
       AAA           29
       BLOCKA         5

STATISTICS
  PROGRAM LENGTH                301B     193
  BUFFER LENGTH                4006B    2054
  CM LABELED COMMON LENGTH       42B      34
  CM BLANK COMMON LENGTH        144B     100
          52000B CM USED
```

Figure 1-12-3. Main Program of Main Overlay (0, 0)

```
      1              FUNCTION TRY(A,B)
                    10 TRY = SQRT(A)+SQRT(B)
                       RETURN
                       ENTRY TRI
      5                IF (A.LE.B) 10,20
                    20 TRY = SQRT(A)-SQRT(B)
                       RETURN
                       END

0
       SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
   14  TRI              4  TRY

VARIABLES     SN  TYPE           RELOCATION
    0  A         REAL                F.P.          0  B         REAL                F.P.
   34  TRY       REAL

EXTERNALS          TYPE   ARGS
       SQRT        REAL      1 LIBRARY

STATEMENT LABELS
    7  10                                  0  20          INACTIVE

STATISTICS
  PROGRAM LENGTH                  35B      29
         52000B CM USED
```

Figure 1-12-4. Function Subroutine of Main Overlay (0, 0)

```
      1              SUBROUTINE PRNT(A)
                     COMMON//D(100)
                     COMMON/AAA/P(29)
                     COMMON/A/SUB(5)
      5              B=0
                     DO 50 I=1,29
                  50 B=B+P(I)
                     PRINT 55,B, (SUB(I),I=1,5)
                  55 FORMAT (1X,6F17.7)
     10              RETURN
                     END

       SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
    3  PRNT

VARIABLES     SN  TYPE           RELOCATION
    0  A         REAL       *UNUSED    F.P.          26  B         REAL
    0  D         REAL        ARRAY     / /           27  I         INTEGER
    0  P         REAL        ARRAY     AAA            0  SUB       REAL       ARRAY    A

FILE NAMES         MODE
       OUTPUT      FMT

STATEMENT LABELS
    0  50                                  24  55       FMT

LOOPS  LABEL     INDEX      FROM-TO     LENGTH     PROPERTIES
   11  50        I            6 7          3B      INSTACK

COMMON BLOCKS    LENGTH
       / /          100
       AAA           29
       A              5

STATISTICS
  PROGRAM LENGTH                  30B      24
  CM LABELED COMMON LENGTH        42B      34
  CM BLANK COMMON LENGTH         144B     100
         52000B CM USED
```

Figure 1-12-5. Subroutine of Main Overlay (0, 0)

```
      1              OVERLAY(OVLA,1,0)
                     PROGRAM OVL10
                     COMMON/AAA/W(29)
                     PRINT 105,(W(I),I=1,7)
      5              PRINT 105,(W(I),I=8,14)
                     PRINT 105,(W(I),I=15,21)
                     PRINT 105,(W(I),I=22,28)
                     PRINT 106,(W(29))
                 105 FORMAT(1X,7F17.7)
     10          106 FORMAT(4X,F17.7)
                     END


       SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
    5  OVL10

VARIABLES     SN  TYPE          RELOCATION
   51  I          INTEGER                          0  W         REAL      ARRAY    AAA

FILE NAMES       MODE
      OUTPUT     FMT

STATEMENT LABELS
   45  105      FMT                          47  106      FMT

COMMON BLOCKS    LENGTH
      AAA           29

STATISTICS
  PROGRAM LENGTH                 52B      42
  CM LABELED COMMON LENGTH       35B      29
         52000B CM USED
```

Figure 1-12-6, Main Program of Primary Overlay (1, 0)

```
OVERLAY(OVL,0,0)

FWA OF THE LOAD            111
LWA+1 OF THE LOAD        12746

TRANSFER ADDRESS -- TESTA          4312

PROGRAM ENTRY POINTS --       TESTA          4312


PROGRAM AND BLOCK ASSIGNMENTS.

BLOCK       ADDRESS   LENGTH    FILE      DATE      PROCSSR VER LEVEL  HARDWARE   COMMENTS

/AAA/           111       35
/BLOCKA/        146        5
TESTA           153     4307    LGO       78/02/16 FTN      4.7 466    666X I     PROGRAM   OPT=1
TRY            4462       35    LGO       78/02/16 FTN      4.7 466    666X I     FUNCTION  OPT=1
/A/            4517        5
PRNT           4524       30    LGO       78/02/16 FTN      4.7 466    666X I     SUBROUTINEOPT=1
SQRT.          4554       32 SL-FORTRAN   78/01/28 COMPASS  3.5 466               COMPUTE THE SQUARE ROOT OF X. OPT=ALL.
SYSAID=        4606        1 SL-FORTRAN   78/01/28 COMPASS  3.5 466               LINK BETWEEN SYS=AID AND INITIALIZATION CO
SYS=1ST        4607       63 SL-FORTRAN   78/01/28 COMPASS  3.5 466               MATH LIBRARY LINK TO ERROR MESSAGE PROCESS
/STP.END/      4672        1
/FCL.C./       4673       25
/Q8.IO./       4720       77
Q2NTRY=        5017        0 SL-FORTRAN   78/01/28 COMPASS  3.5 466               FCL INITIALIZATION ROUTINE.
/FCL=ENT/      5017       40
COMIO=         5057       37 SL-FORTRAN   78/01/28 COMPASS  3.5 466               COMMON CODED I/O ROUTINES AND CONSTANTS.
FCL=FDL        5116       40 SL-FORTRAN   78/01/28 COMPASS  3.5 466               FCL CAPSULE LOADING
FECMSK=        5156       41 SL-FORTRAN   78/01/28 COMPASS  3.5 466               INITIALIZE CONSTANTS.
FLTOUT=        5217      311 SL-FORTRAN   78/01/28 COMPASS  3.5 466               COMMON FLOATING OUTPUT CODE
FMTAP=         5530      356 SL-FORTRAN   78/01/28 COMPASS  3.5 466               CRACK APLIST AND FORMAT FOR KODER/KRAKER.
FORSYS=        6106      271 SL-FORTRAN   78/01/28 COMPASS  3.5 466               FORTRAN OBJECT LIBRARY UTILITIES.
FORUTL=        6377       46 SL-FORTRAN   78/01/28 COMPASS  3.5 466               FCL MISC. UTILITIES.
GETFIT=        6445       57 SL-FORTRAN   78/01/28 COMPASS  3.5 466               LOCATE AN FIT GIVEN A FILE NAME.
KODER=         6524      450 SL-FORTRAN   78/01/28 COMPASS  3.5 466               OUTPUT FORMAT INTERPRETER.
OUTC=          7174      150 SL-FORTRAN   78/01/28 COMPASS  3.5 466               FORMATTED WRITE FORTRAN RECORD.
OUTCOM=        7344      154 SL-FORTRAN   78/01/28 COMPASS  3.5 466               COMMON OUTPUT CODE
OVERLAY        7520      171 SL-FORTRAN   78/01/28 COMPASS  3.5 466               OVERLAY LOADING ROUTINE.
CPUCPM         7711        5 SL-SYSLIB    77/12/13 COMPASS  3.5 466               73/06/12. 77/12/12. CONTROL POINT MANAGERP
CPUMVE         7716       63 SL-SYSLIB    77/12/13 COMPASS  3.5 466               73/06/12. 77/12/12. MOVE BLOCK OF DATA.
CPUSYS        10001       37 SL-SYSLIB    77/12/13 COMPASS  3.5 466               73/06/12. 77/12/12. PROCESS SYSTEM REQUEST
CMF.ALF       10040      160 SL-SYSLIB    78/01/28 COMPASS  3.5 466               CMM V1.1 - ALLOCATE FIXED.
CMM.CIA       10220      106 SL-SYSLIB    78/01/28 COMPASS  3.5 466               CMM V1.1 - CHANGE INTERNAL AREA.
CMF.CSF       10326        6 SL-SYSLIB    78/01/28 COMPASS  3.5 466               CMM V1.1 - CHANGE SPECS FIXED.
CMM.FFA       10334       14 SL-SYSLIB    78/01/28 COMPASS  3.5 466               CMM V1.1 - FIXED FREE ALGORITHM.
CMF.FRF       10350       36 SL-SYSLIB    78/01/28 COMPASS  3.5 466               CMM V1.1 - FREE FIXED.
CMF.LDV       10406      271 SL-SYSLIB    78/01/28 COMPASS  3.5 466               CMM V1.1 - LOAD OVERLAY.
CMF.LOV       10677       55 SL-SYSLIB    78/01/28 COMPASS  3.5 466               CMM V1.1 - LOAD OVERLAY VIA FOL.
CMM.R         10754      213 SL-SYSLIB    78/01/28 COMPASS  3.5 466               CMM V1.1 - RESIDENT SUBROUTINES.
CMF.SLF       11167       22 SL-SYSLIB    78/01/28 COMPASS  3.5 466               CMM V1.1 - SHRINK AT LWA FIXED.
/FDL.COM/     11211       14
FDL.RES       11225      211 SL-SYSLIB    78/01/28 COMPASS  3.5 466               FAST DYNAMIC LOADER RESIDENT.
FDL.MMI       11436      222 SL-SYSLIB    78/01/28 COMPASS  3.5 466               FDL MEMORY MANAGER INTERFACE.
FOL.RES       11660      124 SL-SYSLIB    78/01/28 COMPASS  3.5 466               FAST OVERLAY LOADER RESIDENT.
CTL$RM        12004      432 SL-SYSLIB    78/01/28 COMPASS  3.5 466               CRM CONTROLLING ROUTINE.
CTL$WR        12436       31 SL-SYSLIB    78/01/28 COMPASS  3.5 466               CRM CONTROLLER - WEOX, REWIND
ERR$RM        12467       25 SL-SYSLIB    78/01/28 COMPASS  3.5 466               CRM ERROR PROCESSOR ENTRY.
LIST$RM       12514       66 SL-SYSLIB    78/01/28 COMPASS  3.5 466               CRM - ALLOCATE SPACE FOR LIST OF FILES
 //           12602      144
```

Figure 1-12-7. Loader Map of Main Overlay (0, 0) (Sheet 1 of 4)

```
ENTRY POINTS.

ENTRY      ADDRESS   PROGRAM          REFERENCES

AAM$GO     *UNSAT*                    CTL$RM        12264
WEOI$SQ    *UNSAT*                    CTL$RM        12070
SKIP$SQ    *UNSAT*                    CTL$RM        12064
BACKSP.    *UNSAT*                    FORSYS=        6362
FECPRT$    *UNSAT*                    FCL=ENT        5032
GET$S      *UNSAT*                    CTL$RM        12025
CHEK$SQ    *UNSAT*                    CTL$RM        12011
GET$WA     *UNSAT*                    CTL$RM        12027
GPTM$SQ    *UNSAT*                    CTL$RM        12030
CLOS$RI    *UNSAT*                    FCL=ENT        5022
CLSF$RM    *UNSAT*                    CTL$RM        12013
IOERR$     *UNSAT*                    FCL=ENT        5037
CLSV$SQ    *UNSAT*                    CTL$RM        12014
CMM.AFR    *UNSAT*                    CMM.CIA       10264
CMM.AGR    *UNSAT*                    CTL$RM        12226   12231
OPEN$RM    *UNSAT*                    CTL$RM        12036
LABL$RM    *UNSAT*                    CTL$RM        12031
DBUG.OM    *UNSAT*                    OVERLAY        7647    7651
REPL$SQ    *UNSAT*                    CTL$RM        12042
CMM.CRD    *UNSAT*                    CMM.CIA       10260
DF$CRM     *UNSAT*                    CTL$RM        12022
RPVCOD$    *UNSAT*                    FCL=ENT        5043
RPVCOD.    *UNSAT*                    FCL=ENT        5044
RMP$LR     *UNSAT*                    CTL$RM        12340
CMM.GOA    *UNSAT*                    CMF.ALF       10054   10064
SKFL$SQ    *UNSAT*                    CTL$RM        12063
COMM$WA    *UNSAT*                    CTL$RM        12017
FEIFST.    *UNSAT*                    FCL=ENT        5035
SKSB$SQ    *UNSAT*                    CTL$RM        12065
CMM.PMV    *UNSAT*                    CMM.CIA       10256
CMM.PUA    *UNSAT*                    CMF.LDV       10432   10434
                                      CMF.LOV       10707   10711
CMM.RLS    *UNSAT*                    CMF.LDV       10447
                                      CMF.LOV       10724
CMM.RVR    *UNSAT*                    CMM.CIA       10262
REW$WA     *UNSAT*                    CTL$RM        12044
CMM.SV     *UNSAT*                    CMM.R         11145
CMM.VAF    *UNSAT*                    CMF.ALF       10216
CMM.VFA    *UNSAT*                    CMM.FFA       10342   10343
PUT$WA     *UNSAT*                    CTL$RM        12041
PUT$SQ     *UNSAT*                    CTL$RM        12040
RM$PARI    *UNSAT*                    CTL$RM        12366
CTRL$AA    *UNSAT*                    CTL$RM        12021
GET$SQ     *UNSAT*                    CTL$RM        12026
SYS2$      *UNSAT*                    FCL=ENT        5051
FLEF$RM    *UNSAT*                    CTL$RM        12024
PUT$S      *UNSAT*                    CTL$RM        12037
FEIERR$    *UNSAT*                    FCL=ENT        5034
REW$SQ     *UNSAT*                    CTL$RM        12043
EXP.MSG    *UNSAT*                    SYS=1ST        4634
SYSERR$    *UNSAT*                    FCL=ENT        5047
SKSF$SQ    *UNSAT*                    CTL$RM        12066
RM$PARO    *UNSAT*                    CTL$RM        12374
SKBL$SQ    *UNSAT*                    CTL$RM        12062
OUTPUT#    2227    TESTA              PRNT           4543
TRY        4466    TRY                TESTA          4324
TRI        4476                       TESTA          4330
PRNT       4527    PRNT               TESTA          4316    4346
SQRT.      4570    SQRT.              TRY            4471    4473    4507    4511
SYSAID=    4606    SYSAID=            SYS=1ST        4626
                                      Q8.IO.         4743
SYS1ST.    4612    SYS=1ST            SQRT.          4603
STP.END    4672    Q2NTRY=            FCL=ENT        5045
FCL.C.     4673    FORUTL=            FCL=ENT        5027
                                      COMIO=         5115
Q8.IO.     4720    Q2NTRY=            FCL=ENT        5042
Q2NTRY.    4744                       TESTA          4312
FECCHR.    5063    COMIO=             KODER=         6602    6667
FECBFB.    5103                       KODER=         7157
                                      OUTC=          7272
FECPRT.    5111                       FCL=ENT        5033
                                      FNTAP=         6024
FDLCLL.    5136    FCL=FDL            COMIO=         5115
                                      FORSYS=        6205    6225    6246    6251
```

Figure 1-12-7. Loader Map of Main Overlay (0, 0) (Sheet 2 of 4)

```
FECMSK.   5156  FECMSK=
FEOFAL.   5217  FLTOUT=
FEOEOV.   5241
FEOEXP.   5243
FEORND.   5300
FEOSCA.   5335
FEOZRO.   5422
FECMAP.   5540  FMTAP=
FECAP.    5546
FECFMT.   5565
FECFMU.   5570
FECJP=    5676
FECLP.    5677
FECRP.    5722
FECEE.    5737
FECV.     6006
FECBUG.   6015
END.      6106  FORSYS=
AB1.      6156

IOERR.    6203

SYSEND.   6206
CLSLNK.   6226
SYSERR.   6242

SYS2=     6247
COD.      6252
FECOPE.   6263

CBD.      6402  FORUTL=
BFN.      6407
FAT.      6414

DETA.     6420
DET.      6424

GETFIT.   6451  GETFIT=

KOJPT.    6525  KODER=
KODEND=   7150
KODWRT=   7156
KOREP.    7164
OUTCI.    7211  OUTC=
FEOL.     7344  OUTCOM=
FEOI.     7347
FEOXFL.   7416

FEOAFM.   7424
FEOBLS.   7431

FEOCNV.   7444
FEOR1F.   7475
OVERLAY   7522  OVERLAY
CPM=      7712  CPUCPM
MVE=      7743  CPUMVE

SYS=     10004  CPUSYS

FCL=ENT   5030
COMIO=    5064  5070  5074
FLTOUT=   5227  5227  5244  5353  5372  5373  5401
KODER=    6560  6570  6621  6652  6657  6672  6717
          7001  7036  7041  7054  7065  7115  7151
OUTCOM=   7357  7372  7377  7416  7424  7431  7444
          7444  7466  7475  7502  7507
KODER=    6764  7003  7025
KODER=    7003
KODER=    6765
KODER=    6752  7000  7021
KODER=    6742  6775  7015
KODER=    6753  7001  7023
KODER=    6607  6664  6670  6710  6725  7147  7165
OUTC=     7273
KODER=    6604  7125  7163
KODER=    6543  6647
OUTC=     7265
KODER=    6576
KODER=    6577
KODER=    6730
KODER=    6553
KODER=    6524  6525
TESTA     4351
STP.END   4672
Q8.IO.    5000
FCL=ENT   5040
OUTC=     7330
FCL=ENT   5046
FCL=ENT   5023
Q8.IO.    4736
FCL=ENT   5050
FMTAP=    6043
GETFIT=   6506
OUTC=     7336
OVERLAY   7655
FCL=ENT   5052
FCL=ENT   5024
FCL=ENT   5031
OUTC=     7247
FCL=ENT   5021
FCL=ENT   5020
FCL=ENT   5026
FORSYS=   6160
FORSYS=   6122
FCL=ENT   5025
FORSYS=   6121  6157
FCL=ENT   5036
OUTC=     7217
OUTC=     7207
OUTC=     7274
OUTC=     7261
OUTC=     7210
PRNT      4541
KODER=    6542  6605
KODER=    6537  6540
FLTOUT=   5242  5433
KODER=    7027  7077
FLTOUT=   5266  5271  5273  5277  5431
FLTOUT=   5222  5224  5226  5416  5417  5424  5426
KODER=    6757  7107
FLTOUT=   5240
FLTOUT=   5414
TESTA     4314  4344
CTL$RM   12020
CMM.CIA  10232 10313
CMF.LDV  10424 10530
Q8.IO.    4741  4752
FCL=FDL   5132
FORSYS=   6125  6163
FORUTL=   6430
OVERLAY   7630
CPUCPM    7711
CMM.R    11026 11100 11154
FDL.MMI  11546 11604
FOL.RES  11750 11776
CTL$RM   12067
ERR$RM   12507
```

Figure 1-12-7. Loader Map of Main Overlay (0, 0) (Sheet 3 of 4)

```
MSG=        10032                   Q8.IO.      4750
                                    FCL=ENT     5041
                                    FCL=FDL     5130
                                    FORSYS=     6120    6123    6153    6155    6161
                                    CMM.R      11076   11152
                                    CTL$RM     12034
                                    ERR$RM     12505
CMM.ALF     10040   CMF.ALF         FDL.MMI    11504
                                    CTL$RM     12015   12233   12235
CMM.CIA     10220   CMM.CIA         CMF.LDV    10502
CMM.CSF     10326   CMF.CSF         FDL.MMI    11454   11560
CMM.FFA     10334   CMM.FFA         CMF.FRF    10403
                                    CMF.SLF    11206
CMM.FRF     10350   CMF.FRF         FDL.MMI    11475   11553   11563   11610
                                    CTL$RM     12016
CMM.LDV     10406   CMF.LDV         OVERLAY     7632
CMM.LOV     10677   CMF.LOV         OVERLAY     7621    7622
CMM.AUS     10754   CMM.R           CMM.CIA    10226
                                    CMF.FRF    10357
                                    CMF.SLF    11205
CMM.CFL     10761                   CMF.ALF    10211
                                    CMM.CIA    10276
                                    CMM.FFA    10346
                                    CMF.LDV    10441
                                    CMF.LOV    10715
CMM.CUL     11032                   CMF.ALF    10055
CMM.ICM     11061                   CMF.ALF    10041
                                    CMF.LDV    10417
                                    CMF.LOV    10700
CMM.MEP     11135                   CMF.ALF    10060
                                    CMM.CIA    10274
CMM.SLF     11167   CMF.SLF         FDL.MMI    11453   11557
FDL=PRS     11242   FDL.RES         FDL.MMI    11443   11461
FDL=RCI     11252                   FDL.MMI    11450
FDL=RSC     11342                   FDL.MMI    11457   11467   11471   11471   11476   11534   11564
                                               11573   11611   11655
FDL=SEA     11356                   FDL.MMI    11644
FDL=SPL     11404                   FDL.MMI    11455   11474
FDL.LDC     11442   FDL.MMI         FCL=FDL     5141
                                    CTL$RM     12304   12305
FDL.ULC     11460                   FCL=FDL     5146
                                    CTL$RM     12327   12330
FOL.GDE     11671   FOL.RES         CMF.LOV    10705
FOL.LOV     11714                   OVERLAY     7606    7617
                                    CMF.LOV    10747
LOF$RM      12004   CTL$RM          FCL=FDL     5120
                                    GETFIT=     6475
RM$PL       12010                   CTL$WR     12442   12463   12464
                                    ERR$RM     12473
RM$FOC      12254                   CTL$WR     12453
RM$FAT      12260                   CTL$WR     12437
RM$LGO      12261                   CTL$WR     12443   12463   12465
RM$BLD      12267                   ERR$RM     12474
CLOF$RM     12342                   FCL=FDL     5117
                                    FORSYS=     6234    6311
OPNM$RM     12345                   FORSYS=     6347
PUT$RM      12371                   FCL=FDL     5121
                                    OUTC=       7322
WEOS$RM     12444   CTL$WR          OVERLAY     7573
ERR$RM      12467   ERR$RM          CTL$RM     12023   12234   12256   12257   12260   12313   12314
                                               12322   12405   12424   12434   12435
LIST$RM     12514   LIST$RM         CTL$RM     12004
```

Figure 1-12-7. Loader Map of Main Overlay (0,0) (Sheet 4 of 4)

```
OVERLAY(OVLA,1,0)

FWA OF THE LOAD          12753
LWA+1 OF THE LOAD        13025

TRANSFER ADDRESS -- OVL10          12760


PROGRAM AND BLOCK ASSIGNMENTS.

BLOCK       ADDRESS   LENGTH   FILE     DATE      PROCSSR VER LEVEL  HARDWARE   COMMENTS

OVL10         12753       52   LGO      78/02/16 FTN      4.7 466    666X I     PROGRAM   OPT=1


ENTRY POINTS.

ENTRY     ADDRESS   PROGRAM          REFERENCES

OUTPUT#      2227   TESTA            OVL10          12774      13000      13004      13010      13014
Q2NTRY.      4744   Q2NTRY=          OVL10          12760
END.         6106   FORSTS=          OVL10          12773
OUTCI.       7211   OUTC=            OVL10          12762      12764      12766      12770      12772


      .898 CP SECONDS              33000B CM STORAGE USED                    17 TABLE MOVES
```

Figure 1-12-8. Loader Map of Primary Overlay (1,0)

```
     1.0000000      2.0000000      3.0000000      4.0000000      5.0000000      6.0000000      7.0000000
     8.0000000      9.0000000     10.0000000    100.0000000    200.0000000    300.0000000    400.0000000
   500.0000000    600.0000000    700.0000000    800.0000000    900.0000000   1000.0000000    101.0000000
     2.1000000      3.1000000    202.0000000      2.2000000      3.2000000    303.0000000      2.3000000
     3.3000000
  6177.2000000      0.0000000      0.0000000      0.0000000      0.0000000      0.0000000
     2.0000000      4.0000000      6.0000000      8.0000000     10.0000000     12.0000000     14.0000000
    16.0000000     18.0000000     20.0000000      2.8284271      4.0000000      4.8989795      5.6568542
     6.3245553      6.9282032      7.4833148      8.0000000      8.4852814      8.9442719    101.0000000
     2.1000000      3.1000000    202.0000000      2.2000000      3.2000000    303.0000000      2.3000000
     3.3000000
   795.7498875      0.0000000      0.0000000      0.0000000      0.0000000      0.0000000
     1.0000000      2.0000000      3.0000000      4.0000000      5.0000000      6.0000000      7.0000000
     8.0000000      9.0000000     10.0000000    100.0000000    200.0000000    300.0000000    400.0000000
   500.0000000    600.0000000    700.0000000    800.0000000    900.0000000   1000.0000000    101.0000000
     2.1000000      3.1000000    202.0000000      2.2000000      3.2000000    303.0000000      2.3000000
     3.3000000
  6177.2000000      0.0000000      0.0000000      0.0000000      0.0000000      0.0000000
     2.0000000      4.0000000      6.0000000      8.0000000     10.0000000     12.0000000     14.0000000
    16.0000000     18.0000000     20.0000000      2.8284271      4.0000000      4.8989795      5.6568542
     6.3245553      6.9282032      7.4833148      8.0000000      8.4852814      8.9442719    101.0000000
     2.1000000      3.1000000    202.0000000      2.2000000      3.2000000    303.0000000      2.3000000
     3.3000000
   795.7498875      0.0000000      0.0000000      0.0000000      0.0000000      0.0000000
```

Figure 1-12-9. Program Output

```
1    EXCHANGE PACKAGE.
R
     P     10004  A0    2227  B0       0      (A0)  1725 2420 2524 0000 0131
     RA   742400  A1       1  B1       1      (A1)  0516 0420 0000 0000 0000
     FL    15100  A2    6444  B2  777755      (A2)  1717 0631 4631 4640 3615
     EM     7007  A3   15020  B3    6235      (A3)  6000 0000 0131 5700 0000
     RAE   10000  A4    6402  B4   11460      (A4)  0400 0064 4100 0000 0000
     FLE    7000  A5    4714  B5   13157      (A5)  2000 0000 0000 0001 1201
     MA     4000  A6       1  B6    2000      (A6)  0516 0420 0000 0000 0000
                  A7   14217  B7      30      (A7)  0000 0000 0131 5701 4726

     X0   0000 0000 0000 0000 0000
     X1   0000 0000 0000 0000 0000
     X2   1717 0631 4631 4640 3615
     X3   2000 0000 0000 0000 0012
     X4   2000 0000 0000 0000 0000
     X5   0000 0000 0000 0000 0001
     X6   0516 0420 0000 0000 0000
     X7   2000 0000 0000 0000 0001

   (RA)   0000 0000 0000 0000 0000
   (RA+1) 0516 0420 0000 0000 0000


  CM DUMP FROM    7744 TO   10044.
R
    7744      37423 10711 10100 20236    03070 07743 72677 76314    10044 03340 07747 14044    36007 11606 03260 07752
    7750      10077 12223 13777 43470    61700 07772 02000 07774    71600 07717 03340 07763    12226 20636 12336 46000
    7754      10077 72777 77753 43470    03370 07756 71000 00024    61700 07757 02000 07774    46400 07716 61000 46000
    7760      63770 10011 07710 07743    71407 77753 37334 20436    37224 02000 07754 46000    20636 12676 36336 20636
    7764      36226 46000 61000 46000    71600 00024 73070 37776    43470 03370 07767 10066    10600 37330 20636 37226
    7770      61700 07771 02000 07774    46400 07716 61000 46000    63770 10011 07710 07743    02000 07765 61000 46000
    7774      22610 20003 36606 15064    11446 20032 20454 12040    12620 51600 07716 12630    46400 07716 61000 46000
   10000      54660 02770 00000 46000    54110 20123 03310 10003    04000 10014 00000 00000    01300 00000 61000 46000
   10004      04000 06126 00000 00000    51100 00001 03110 10005    54610 04000 10003 46000    51100 10001 10611 46000
   10010      51100 00066 03210 10012    51100 10002 10611 46000    51600 10003 51100 00001    10611 01000 10002 46000
   10014      20652 01000 10004 46000    51100 00001 03110 10015    00000 00000 61000 46000    51100 00001 03110 10016
   10020      71602 20314 04000 10014    71102 20314 27601 20652    12662 01000 10004 46000    00000 00000 61000 46000
   10024      53120 20173 03310 10023    03010 10023 51100 00001    03010 10021 04000 10024    20151 46000 61000 46000
   10030      71601 52307 20652 12661    01000 10004 61000 46000    04000 06124 00000 00000    20630 12161 73610 20123
   10034      03210 10027 20151 13116    20636 51600 10037 74660    12116 04000 10030 46000    00000 06127 00000 00001
   10040      04000 11505 00000 00000    61100 00001 01000 11061    71600 11406 50607 77746    51100 10040 20136 10611
   10044      50607 77747 10622 43766    50607 77744 21306 15737    21306 73630 50607 77741    50707 77740 73721 46000
```

Figure 1-12-10. Exchange Package Dump

```
 CM DUMP FROM       0 TO    4500.

      0      00000 00000 00000 00000    05160 42000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000
      4      00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000
         DUPLICATED LINES.
     54      56110 03110 00054 54710    51100 00001 03110 00055    64550 02550 00000 46000    00000 00000 00000 00000
     60      15051 52000 00000 00061    00000 13100 00000 00001    00000 00000 00000 00000    00000 00000 00000 00000
     64      17261 40000 00000 00000    40000 00000 00007 64752    40000 00000 01000 12746    40000 00000 40000 00000
     70      17261 45700 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000
     74      00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000
    100      54000 00000 01000 00001    01247 10000 00000 12746    00000 00000 00000 00000    00000 00000 00000 00000
    104      00000 00000 00000 13025    00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000
    110      24052 32401 00000 04312    17214 00000 00000 00000    17224 00000 00000 00000    17226 00000 00000 00000
    114      17234 00000 00000 00000    17235 00000 00000 00000    17236 00000 00000 00000    17237 00000 00000 00000
    120      17244 00000 00000 00000    17244 40000 00000 00000    17245 00000 00000 00000    17215 52023 63147 74736
    124      17224 00000 00000 00000    17224 71421 60500 44411    17225 52023 63147 74736    17226 24613 01655 51333
    130      17226 73317 27205 41145    17227 36735 20426 40772    17234 00000 00000 00000    17234 17416 66315 75547
    134      17234 36156 74671 37645    17226 00000 00000 00000    17214 14631 46314 63146    17216 14631 46314 63146
    140      17276 24000 00000 00000    17214 31463 14631 46315    17216 31463 14631 46315    17304 57000 00000 00000
    144      17214 46314 63146 31463    17216 46314 63146 31463    17226 00000 00000 00000    17236 00000 00000 00000
    150      17244 40000 00000 00000    17246 00000 00000 00000    17247 40000 00000 00000    11162 02524 00000 00001
    154      00000 00000 00000 00224    00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000
    160      00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000
    164      00000 00000 00000 00000    00000 00000 00600 00000    00000 00022 00000 00000    00000 00000 00000 00000
    170      14000 00000 00000 00000    00000 00000 00000 00000    00000 00006 00000 00000    00000 00000 00000 00000
    174      00000 00000 00000 02003    00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000
    200      00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000
         DUPLICATED LINES.
   2224      00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000    17252 42025 24000 00131
   2230      04130 46000 00360 02300    00000 00000 00000 02300    00000 00000 00000 02300    63360 00000 01000 02302
   2234      00000 00000 00000 00000    00000 02237 00000 00000    00000 00000 00000 00000    00000 00000 00000 00001
   2240      00000 00000 00000 00000    00000 00004 00600 00000    00000 14762 03200 00000    00000 22600 00000 00000
   2244      14000 00000 00000 06120    00000 00000 00000 00000    00000 00006 00000 04602    00000 00000 00000 03720
   2250      23000 00000 00000 02003    00002 27002 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000
   2254      00000 00000 00000 00000    00004 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000
```

Figure 1-12-11. Central Memory Dump (Sheet 1 of 2)

```
2260     00000 00000 00000 00000    00000 00000 00000 00000    00000 10000 00000 00000    00000 00000 00000 00000
2264     00000 00000 00000 00000    01040 02300 00000 00000    00000 00000 00000 00226    00000 00000 00000 00000
2270     00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000
     DUPLICATED LINES.
2300     55555 55555 55555 55534    57333 33333 33333 35555    55555 55555 55355 73333    33333 33333 55555 55555
2304     55555 53657 33333 33333    33335 55555 55555 55555    37573 33333 33333 33355    55555 55555 55554 05733
2310     33333 33333 33555 55555    55555 55541 57333 33333    33333 35555 55555 55555    55425 73333 33333 33333
2314     00000 00000 00000 00000    55555 55555 55555 55543    57333 33333 33333 35555    55555 55555 55445 73333
2320     33333 33333 55555 55555    55553 43357 33333 33333    33335 55555 55555 53433    33573 33333 33333 33355
2324     55555 55555 35333 35733    33333 33333 33555 55555    55553 63333 57333 33333    33333 35555 55555 55537
2330     33335 73333 33333 33333    00000 00000 00000 00000    55555 55555 55554 03333    57333 33333 33333 35555
2334     55555 55541 33335 73333    33333 33333 55555 55555    55423 33357 33333 33333    33335 55555 55555 54333
2340     33573 33333 33333 33355    55555 55555 44333 35733    33333 33333 33555 55555    55343 33333 57333 33333
2344     33333 35555 55555 55534    33345 73333 33333 33333    00000 00000 00000 00000    55555 55555 55555 55535
2350     57343 33333 33333 35555    55555 55555 55365 73433    33333 33333 55555 55555    55353 33557 33333 33333
2354     33335 55555 55555 55555    35573 53333 33333 33355    55555 55555 55553 65735    33333 33333 33555 55555
2360     55553 63336 57333 33333    33333 35555 55555 55555    55355 73633 33333 33333    00000 00000 00000 00000
2364     55555 55555 55555 55555    55553 65736 33333 33333    33000 00000 00000 00000    55555 55555 55413 44242
2370     57353 33333 33333 35555    55555 55555 55335 73333    33333 33333 55555 55555    55555 53357 33333 33333
2374     33335 55555 55555 55555    33573 33333 33333 33355    55555 55555 55553 35733    33333 33333 33555 55555
2400     55555 55533 57333 33333    33333 30000 00000 00000    55555 55555 55555 55535    57333 33333 33333 35555
2404     55555 55555 55375 73333    33333 33333 55555 55555    55555 54157 33333 33333    33335 55555 55555 55555
2410     43573 33333 33333 33355    55555 55555 55343 35733    33333 33333 33555 55555    55555 53435 57333 33333
2414     33333 35555 55555 55555    34375 73333 33333 33333    00000 00000 00000 00000    55555 55555 55555 53441
2420     57333 33333 33333 35555    55555 55555 34435 73333    33333 33333 55555 55555    55553 53357 33333 33333
2424     33335 55555 55555 55555    35574 33543 37354 23455    55555 55555 55553 75733    33333 33333 33555 55555
2430     55555 55537 57434 44344    42444 05555 55555 55555    55405 74140 41434 03735    00000 00000 00000 00000
2434     55555 55555 55555 55541    57363 53740 40403 65555    55555 55555 55415 74435    43353 33635 55555 55555
2440     55555 54257 37433 63634    37435 55555 55555 55555    43573 33333 33333 33355    55555 55555 55554 35737
2444     43403 54334 37555 55555    55555 55543 57443 73735    42344 45555 55555 55534    33345 73333 33333 33333
2450     00000 00000 00000 00000    55555 55555 55555 55535    57343 33333 33333 35555    55555 55555 55365 73433
2454     33333 33333 55555 55555    55353 33557 33333 33333    33335 55555 55555 55555    35573 53333 33333 33355
2460     55555 55555 55553 65735    33333 33333 33555 55555    55553 63336 57333 33333    33333 35555 55555 55555
2464     55355 73633 33333 33333    00000 00000 00000 00000    55555 55555 55555 55555    55553 65736 33333 33333
2470     33000 00000 00000 00000    55555 55555 55554 24440    57423 74443 43424 05555    55555 55555 55335 73333
2474     33333 33333 55555 55555    55555 53357 33333 33333    33335 55555 55555 55555    33573 33333 33333 33355
2500     55555 55555 55553 35733    33333 33333 33555 55555    55555 55533 57333 33333    33333 30000 00000 00000
2504     00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000
     DUPLICATED LINES.
4300     00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000    20020 00000 00000 04306
4304     00000 00000 00000 11610    00000 00000 00000 00000    11162 02524 00000 00153    17252 42025 24000 02227
4310     00000 00000 00000 00000    24052 32401 55550 04312    51100 04303 01000 04626    51100 04352 46000 46000
4314     01000 07404 00160 04311    51100 04356 46000 46000    01000 04527 00170 04311    71700 00001 51700 04372
4320     51500 04372 52450 00110    72750 00110 40644 46000    51700 04360 51700 04361    54640 51100 04360 46000
4324     01000 04466 00220 04311    51500 04372 72750 00110    52650 00110 51700 04360    51700 04361 51100 04360
4330     01000 04476 00230 04311    51500 04372 72750 00001    72077 77764 52650 00122    54750 03300 04320 46000
4334     71700 00001 51700 04373    51500 04373 61600 00005    36055 63750 62500 00110    56550 51470 00110 30054
4340     61550 00002 24700 46000    51770 00145 61770 00001    06670 04337 76770 46000    51700 04373 51100 04363
4344     01000 07404 00260 04311    51100 04356 46000 46000    01000 04527 00270 04311    51500 04373 51100 04311
4350     27005 24700 51700 00135    04000 05770 46000 46000    00000 00000 00000 04456    00000 00000 00000 04367
4354     00000 00000 00000 04370    00000 00000 00000 00000    00000 00000 00000 04371    00000 00000 00000 00000
4360     00000 00000 00000 00122    00000 00000 00000 00122    00000 00000 00000 00000    00000 00000 00000 04460
4364     00000 00000 00000 04367    00000 00000 00000 04370    00000 00000 00000 00000    00000 00000 00000 00001
4370     00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00013    00000 00000 00000 00006
4374     00000 00000 00000 00173    00000 00000 00000 00173    00000 00000 00000 00173    00000 00000 00000 00173
     DUPLICATED LINES.
4454     00000 00000 00000 00173    00000 00000 00000 00173    17261 40155 55555 55555    00000 00000 00000 00000
4460     17261 40155 55555 55555    00000 00000 00000 00000    24223 15555 55550 04466    00000 00000 00000 13100
4464     51400 04516 10644 46000    51300 04463 52030 00000    04000 04331 00000 00000    74600 54010 51600 04463
4470     46000 46000 46000 46000    54500 53150 01000 07607    50500 00001 53150 46000    51600 04514 01000 07607
4474     51500 04514 30056 24700    51700 04516 04000 04464    04000 04331 00000 00000    51200 04501 10622 46000
4500     51600 04470 04000 04467    04000 04502 61000 46000    51100 04515 51200 04476    10611 22702 51600 04470
```

Figure 1-12-11. Central Memory Dump (Sheet 2 of 2)

SYSTEM UTILITY CONTROL STATEMENTS 13

NOS provides the following utilities for file maintenance.

EDIT	Edits a text file.
KRONREF	Generates a cross-reference listing of system symbols.
MODIFY	Edits a Modify-formatted program library file.
OPLEDIT	Removes modification decks and identifiers from a Modify-formatted program library file.
PROFILE	Enables a master user to update and inquire about a profile file.
UPDATE	Edits an Update-formatted program library file.
UPMOD	Converts an Update-formatted program library file to a Modify-formatted program library file.
XEDIT	Edits a text file.

EDIT STATEMENT

The EDIT control statement calls the Text Editor utility. The Text Editor enables a user to manipulate data on a specified mass storage file through use of special input directives called edit commands. For a detailed description of the Text Editor and an explanation of these commands, refer to the Text Editor Reference Manual.

The control statement format is:

EDIT(lfn_1, m, lfn_2, lfn_3)

or

EDIT(FN=lfn_1, M=m, I=lfn_2, L=lfn_3)

lfn_1	Name of file to be edited (referred to as edit file). This specification is required for batch origin jobs.
m	Mode of file processing: ASCII or AS — ASCII mode edit file. NORMAL or N — NORMAL mode edit file.
lfn_2	File from which directives (edit commands) are to be read. If omitted, INPUT is assumed.
lfn_3	File to which output is to be written. If omitted, OUTPUT is assumed.

KRONREF STATEMENT

The KRONREF control statement generates a cross-reference listing of system symbols used by decks on a Modify OPL.

The control statement format is:

KRONREF(P=lfn_1, L=lfn_2, S=lfn_3, G=lfn_4)

P=lfn_1	OPL input from file lfn_1. If the P option is omitted or P alone is specified, file OPL is assumed.
L=lfn_2	List output on file lfn_2. If the L option is omitted or L alone is specified, file OUTPUT is assumed.
S=lfn_3	System text from overlay lfn_3. If the S option is omitted or S alone is specified, file NOSTEXT is assumed.
G=lfn_4	System text from local file lfn_4. If G is omitted, system text is acquired as specified or defaulted by the S option. If G alone is specified, local file TEXT is used. Use of the G option overrides any S specification.

The names of programs on the OPL are listed for those decks that reference the following.

- PP direct cell locations defined in lfn_3 or lfn_4.
- PP resident entry points defined in lfn_3.
- Monitor functions.
- Central memory pointers (in low core) defined in lfn_3 or lfn_4.
- Central memory locations (in low core) defined in lfn_3 or lfn_4.
- Control point area words defined in lfn_3 or lfn_4.
- Dayfile message options.
- File types and mass storage constants.
- Job origin types, queue types, and priorities.
- Error flags referenced.
- Common deck calls.
- PP packages called.
- Special entry points.

MODIFY STATEMENT

The MODIFY control statement edits a Modify-formatted program library file.

The control statement format is:

MODIFY($p_1, p_2, \ldots, p_n$)

p_i Any of the following in any order:

I	Use directive input from file INPUT. If the I option is omitted, file INPUT is assumed.
I=lfn_1	Use directive input from file lfn_1.
I=0	Use no directive input.
P	Use file OPL for the old program library. If the P option is omitted, file OPL is assumed.
P=lfn_2	Use file lfn_2 for the old program library.
P=0	Use no old program library
C	Write compile output to file COMPILE. If the C option is omitted, file COMPILE is assumed.
C=lfn_3	Write compile output to file lfn_3.
C=0	Write no compile output.
N	Write new program library on file NPL.
N=lfn_4	Write new program on file lfn_4.
N=0	Write no new program library. If this option is omitted, N=0 is assumed.
S	Write source output on file SOURCE.
S=lfn_5	Write source output on file lfn_5.
S=0	Write no source output. If this option is omitted, S=0 is assumed.
L	List output on file OUTPUT. If the L option is omitted, file OUTPUT is assumed.
L=lfn_6	List output on file lfn_6.
L=0	List no output.
LO	Select list options: ECTMWDS
LO=chars	Select up to seven list options which can be any of the following.

- E Errors.
- C Directives other than INSERT, DELETE, RESTORE.
- T Input text.
- M Modifications made.
- W Compile file directives.
- D Deck status.
- S Statistics.
- I Inactive statements.
- A Active statements.

A	Write compressed compile file.
D	Ignore errors.
F	Modify all decks.
U	Modify only decks mentioned on DECK directives; F overrides the U option.
NR	Do not rewind the compile file.
X	Rewind input and output files, set A option, and call the COMPASS assembler when modification is complete.
X=prog	Rewind input and output files, set A option, and call the processing program prog when modification is complete.
X=0	Do not call another processing program. If this option is omitted, X=0 is assumed.
Q	Rewind the output file, set A option, and call the COMPASS assembler when modification is complete.
Q=prog	Rewind the output file, set A option, and call the prog assembler when modification is complete.
Q=0	Do not call another processing program. If this option is omitted, Q=0 is assumed.
Z	If this parameter is present, the MODIFY control card contains the input directives following the terminator. When this parameter is specified, the I parameter is ignored.

NOTE

Do not place another terminator after the directives.

CV=63	Convert 64-character set OPL to 63-character set OPL.
CV=64	Convert 63-character set OPL to 64-character set OPL.

The following parameters are effective only if the X or Q option is selected.

CB	Set assembler argument B=LGO. If the CB option is omitted, B=LGO is assumed.
CB=lfn_7	Set assembler argument B=lfn_7.
CB=0	Set assembler argument B=0.
CL	Set assembler argument L=OUTPUT.
CL=lfn_8	Set assembler argument L=lfn_8.
CL=0	Set assembler argument L=0. If this option is omitted, L=0 is assumed.
CS	Set assembler argument S=SYSTEXT. If the CS option is omitted, S=SYSTEXT is assumed.
CS=lfn_9	Set assembler argument S=lfn_9. †
CS=0	Set assembler argument S=0.
CG	Set assembler argument G=SYSTEXT.
CG=lfn_{10}	Set assembler argument G=lfn_{10}. ††
CG=0	Set assembler argument G=0. If this option is omitted, CG is defined by the CS option.

For a more detailed description of Modify, refer to the Modify Reference Manual.

OPLEDIT STATEMENT

The OPLEDIT control statement removes modification decks and identifiers from a Modify-formatted program library file.

The control statement format is:

OPLEDIT($p_1, p_2, \ldots, p_n$)

p_i Any of the following in any order:

I	Use directive input from file INPUT. If the I option is omitted, file INPUT is assumed.
I=lfn_1	Use directive input from file lfn_1.
I=0	Use no directive input.
P	Use file OPL for the old program library. If the P option is omitted, file OPL is assumed.
P=lfn_2	Use file lfn_2 for the old program library.
P=0	Use no old program library.
N	Write new program library on file NPL.
N=lfn_3	Write new program library on file lfn_3.
N=0	Write no new program library. If this option is omitted, N=0 is assumed.

† The desired file is retrieved from the system.
†† The desired file is a local file.

L	List output on file OUTPUT. If the L option is omitted, file OUTPUT is assumed.
L=lfn_4	List output on file lfn_4.
L=0	List no output.
M=lfn_5	Write output from *PULLMOD directives on file lfn_5. If this option is omitted, M=MODSETS is assumed.
LO=x	Set list options x; each bit in x, if set, turns on the corresponding option. 001 Errors. 002 Input directives. 010 Modifications made. 040 Deck status. 100 Directory lists. If this option is omitted, x=153 is assumed (that is, all options listed).
F	Modify all decks.
D	Debug; ignore errors.
U	Generate *EDIT directives for all decks.
U=0	Generate no *EDIT directives. If the U option is omitted, generate *EDIT directives for common decks.
Z	The OPLEDIT control statement contains the input directives following the terminator; the input file is not read. This eliminates the need to use a separate input file for the directives when only a few directives are needed. The first character following the control statement terminator is the separator character. If Z is omitted, the control statement does not contain the input directives. **NOTE** Do not place another terminator after the directives.

For a complete description of the OPLEDIT utility, refer to the Modify Reference Manual.

PROFILE STATEMENT

The PROFILE control statement enables the master user to update and inquire about a project profile file for user profile control. Other capabilities of PROFILE (available only to system origin jobs) are described in the NOS System Maintenance Reference Manual.

The control statement format is:

PROFILE($p_1, p_2, \ldots, p_n$)

p_i Any of the following in any order:

I=lfn_1 File lfn_1 contains input directives for an update (OP=U). If omitted, file INPUT is assumed.

L=lfn_2 File lfn_2 receives output listings. If omitted, OUTPUT is assumed.

P=lfn_3 File lfn_3 is the project profile file. If omitted, file PROFILB is assumed.

CN=cnum Charge number inquire. All project numbers valid for charge number cnum are written to output file. Valid only if OP=I option specified.

PN=pnum Project number inquire. The control values and all valid user numbers for project number pnum are written to the output file. The OP=I and CN=cnum options must also be specified to use the PN option.

CV Convert option. Specifies that directives on the input file are in NOS 1.0 or 1.1 format and are to be converted to update an NOS 1.2, 1.3 or 1.4 profile file. Obsolete directives are ignored. OP=U or OP=T option must also be specified.

OP=optn PROFILE processing option. One of the following:

optn	Description
L	Lists portions of the profile file as specified by the LO parameter.
U	Updates the project profile file with directives supplied by the input file. U is the default value for the master user if the OP option is omitted.
T	Time-sharing update. Processing is the same as OP=U, but preliminary instructions are suppressed.
I	Inquire option. Output is dependent upon CN and/or PN options specified.

LO=*l*sop PROFILE list option (OP=L must be specified). One of the following:

*l*sop	Description
FM	Full list of everything accessible on the profile file by the master user. This is the default for a nonsystem origin job if the LO parameter is omitted.

CM	Charge number list of all charge numbers accessible on the profile file by the master user.
PM	Project number list of all project numbers accessible on the profile file by the master user.

Directives are available to the master user as input to PROFILE to add or update information concerning each charge number. The input file for a PROFILE update (OP=U) is divided into groups of entries that each begin with a charge number directive. Each directive following a particular charge number entry applies only to that charge number, until another charge number entry occurs.

Each line of the input file can contain one or more directives (up to 72 characters per line) in the following format.

$dir_1, dir_2, \ldots, dir_n$

Each directive is separated by a delimiter which can be any special character (display code greater than 44_8) except the following.

/ + - * :

An end-of-line or end-of-card also delimits directives. The following directives are available to the master user for PROFILE input.

dir_i	Description
/cnum	Specifies the charge number cnum to which the following directives apply. If this form of charge number specification is used (refer to CN=cnum directive), it must begin in column 1 with the slash (/).
APN=pnum	Adds or activates project number pnum.
AUN=un	Adds user number un (must be preceded by PN directive).
CN=cnum	Same as /cnum except that it can begin in any column.
DPN=pnum	Deactivates project number pnum. This directive does not delete the specified project number entry but sets its status such that it cannot be specified by users.
DUN=un	Deletes user number un from the list of those who may access the project number (must be preceded by PN directive).
ISV=x	Sets x as the maximum SRU accumulation for any job using the charge number and project number specified by preceding CN and PN directives.
PEX=yymmdd.	Specifies expiration date for project number of preceding PN directive. If PEX=0 is entered, the project number is not limited by an expiration date.
PN=pnum	Project number for which the following directives (until the next PN directive) apply.
SMA=acc	Sets the current number of accumulated SRUs the project number has used (PN directive required). This accumulator is updated at the end of a job or terminal session and each time a CHARGE control statement is entered. When the SMA value surpasses the SML value, the project is not available to users until either the limit or accumulator is respecified.

dir_i	Description
SML=lim	Specifies the maximum number of accumulated SRUs the project (PN directive) may use. SML=0 implies no restriction.
TI=ti	Specifies the time of day before which the project number specified by the PN directive cannot be used. The time is specified in 24-hour clock notation. For example, a ti specification of 1315 indicates the project number cannot be used before 1:15 in the afternoon.
TO=to	Specifies the time of day before which the project number specified by the PN directive cannot be used. The time is specified in 24-hour clock notation. For example, a ti specification of 1315 indicates that the project number cannot be used before 1:15 in the afternoon.

UPDATE STATEMENT

The UPDATE control statement edits an Update-formatted program library file.

The control statement format is:

UPDATE($p_1, p_2, \ldots, p_n$)

p_i Any of the following in any order:

A	Sequential-to-random program library copy
B	Random-to-sequential program library copy
C	Write compile file output on COMPILE. If the C option is omitted, file COMPILE is assumed.
$C=lfn_1$	Write compile file output on lfn_1.
C=0	Write no compile output.
D	Compile output has 80 columns for data; if D is omitted, compile output has 72 columns for data.
E	Update rearranges the directory to reflect the actual order of decks on the program library. If E is omitted, the old program library directory is not edited.
F	Full update; all decks are compiled. If F is omitted, corrected decks and those named on COMPILE directives are processed.
$G=lfn_2$	Output from PULLMOD directives is written on lfn_2. Any rewind option applying to the source file also applies to this file. OUTPUT is not a valid file for this option. If G is omitted, pulled modifications are appended to the source file.
I	Input is on file INPUT. If the I option is omitted, file INPUT is assumed.
$I=lfn_3$	Input comprises next record on lfn_3.

K	Compile output decks to be written on file COMPILE in COMPILE directive sequence.
K=lfn_4	Compile output decks to be written on lfn_4 in COMPILE directive sequence. If this option is omitted, output is determined by the C option.
L=char	char is a string that specifies any of the A, F, and 0 through 9 list options. If this option is omitted, options A, 1, 2, 3, and 4 are selected. Any use of 0 suppresses listing.
M	Merge input is on file MERGE.
M=lfn_5	Merge input is on file lfn_5. If M option is omitted, there is no merge file.
N	New program library to be written on file NEWPL.
N=lfn_6	New program library to be written on file lfn_6. If N option is omitted, no new program library is written.
O	List output to be written on OUTPUT. If the O option is omitted, OUTPUT is assumed.
O=lfn_7	List output to be written on lfn_7. If O option is omitted, OUTPUT is assumed.
P	Use file OLDPL for the old program library. If the P option is omitted, OLDPL is assumed.
P=lfn_8	Use file lfn_8 for the old program library. If this option is omitted, OLDPL is assumed.
Q	Only decks on COMPILE directives are processed. If Q is omitted, corrected decks and those named on COMPILE directives are processed.
R	No rewinds are issued for the program libraries, compile file, or source file.
R=char	Each character in the string char indicates a file to be rewound before and after the Update run. C Compile. N New program library. P Old program library and merge library. S Source and PULLMOD. Files not specified in char are not rewound. If R is omitted, files are rewound before and after the Update run.
S	Source output written on file SOURCE.
S=lfn_9	Source output written on file lfn_9. If S is omitted, Update does not generate a source output file unless the source output is specified by T.

T	Source output excluding common decks on file SOURCE.
$T=lfn_{10}$	Source output excluding common decks on file lfn_{10}. If T is omitted, no source output unless source output is specified by S.
U	Update execution is not terminated by normally fatal errors. If U is omitted, Update execution terminates upon encountering a fatal error.
W	The new program library (refer to N option) is a sequential file. If W is omitted, the new program library is a random file (unless it is a magnetic tape file).
X	Compile file is in compressed format. If X is omitted, the compile file is not in compressed format.
8	Compile file output is composed of 80-column line images. If this option is omitted, compile file output is composed of 90-column line images.
*=char	The master control character (first character of each directive) for this Update run is char which can be any character having a display code octal value in the range 01 through 54 except for 51 and 52 (the open and close parentheses). If this option is omitted, the master control character is *.
/=char	The comment control character for this Update run is char which can be A through Z, 0 through 9, or +-*/$=. The character should not be changed to one of the abbreviated forms of directives unless NOABBREV is in effect. If this option is omitted, the comment control character is a slant bar.

The UPDATE control statement is processed in product set format. For a more detailed description of Update, refer to the Update Reference Manual.

UPMOD STATEMENT

The UPMOD control statement converts an Update-formatted program library file to a Modify-formatted program library file.

The control statement format is:

UPMOD($p_1, p_2, \ldots, p_n$)

p_i Any of the following in any order:

P	Update program library from file OLDPL. If the P option is omitted, file OLDPL is assumed.
P=lfn_1	Update program library from file lfn_1.
N	Modify program library on file OPL.
N=lfn_2	Modify program library on file lfn_2.
M	Modify program library name is OPL. If the M option is omitted, file OPL is assumed.
M=lfn_3	Modify program library name is lfn_3.
F	Convert to file mark.
NR	Do not rewind file lfn_1.

The Update file must be in sequential format. A random Update file must first be changed to sequential format via Update before being submitted to UPMOD for conversion. Unless otherwise specified, only one record from the Update file is converted. After the Modify OPL has been created, no references should be made to modset identifiers present on the Update library. The new OPL should be treated as any other program library created by a Modify creation run.

XEDIT STATEMENT

The XEDIT control statement calls the text editor, XEDIT. For a complete description of XEDIT parameters and commands, refer to the XEDIT Reference Manual.

The control statement format is:

XEDIT(lfn_1,p_1,p_2,...,p_n)delimited command sequence

All parameters are optional. The following are brief parameter descriptions.

lfn_1 — Name of the file to be edited. If lfn_1 is omitted (indicated by two separators before other parameters), the primary file is edited.

p_i — One or more of the following parameters in any order.

- AS — Edit lfn_1 in ASCII time-sharing mode. After the XEDIT job step, processing returns to the original mode.
- B — Process the job as a batch origin job.
- C — Create a new file lfn_1.
- FR — Take the first editing command from the first line of lfn_1.
- I=lfn_2 — Take editing directives from lfn_2. If I is omitted, commands are taken from file INPUT.
- I=0 — Take all editing directives from the trailing delimited command sequence. (If the FR parameter is also specified, process the delimited command sequence after processing directives from the first line of lfn_1.)
- L=lfn_3 — Put all XEDIT output on the specified file.
- L=0 — Suppress all output.
- NH — Suppress printing of the XEDIT header.
- P — Retrieve and edit permanent file lfn_i. Direct access files are attached in write mode. If P is omitted, the file lfn_1 is assumed to be a local file.

delimited command sequence — Command sequence with the following format (; represents the delimiter character).

;$command_1$;$command_2$;...;$command_n$

LIBRARY MAINTENANCE 14

A library is a file containing records that are accessed individually. Library records can be of several types and can be accessed randomly or sequentially.

This section describes library access methods, library record types, and the following control statements and their functions.

CATALOG	Describes the records on a file.
GTR	Appends records selected from one file to another file.
LIBEDIT	Generates a file containing records from one or more other files. The records may be of several types. LIBEDIT handles a LIBGEN-generated library as a single record.
LIBGEN	Generates a user library, that is, a library of relocatable and capsule records that can be accessed by CYBER Loader.
VFYLIB	Compares the records in two files and lists their differences.

FILE ACCESS METHODS

The methods used to access records within NOS files are sequential access and random access.

To access a record sequentially, NOS rewinds the file to BOI and then reads records until it finds the requested one. To replace, insert, or delete records from a sequential access file, NOS must rewrite the entire file. (Records can be appended at EOI without rewriting the file.)

To access a record randomly, the file must be on mass storage and must have a directory containing the address of each record (figure 1-14-1). The GTR, LIBGEN, and LIBEDIT statements can generate random access directories. Records within a file can then be replaced, inserted, or deleted by rewriting the directory instead of rewriting the entire file. Records within a random access file can also be accessed sequentially.

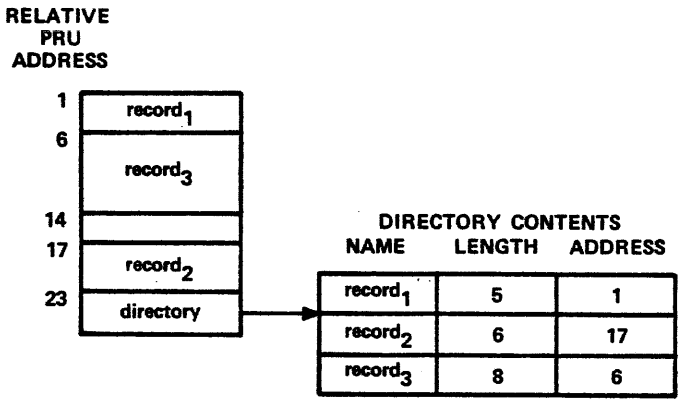


Figure 1-14-1. Random Access File Structure

LIBRARY RECORD TYPES

The following record types can be specified on the GTR and LIBEDIT control statements and are recognized by the CATALOG and VFYLIB statements. LIBGEN generates a user library from relocatable (REL) and capsule (CAP) records.

Mnemonic	Meaning
ABS	Multiple entry point overlay.
CAP	CYBER Loader capsule.
COS	Chippewa format CPU program; COMPASS or FORTRAN source with errors that suppressed binaries.
OPL	Modify old program library deck.
OPLC	Modify old program library common deck.
OPLD	Modify old program library directory.
OVL	Central processor overlay.
PP	Peripheral processor program.
PPU	Peripheral processor unit program.
PROC	CCL procedure file.
REL	Relocatable central processor program.
TEXT	Unrecognizable as a program.
ULIB	User library directory.

Appendix G in volume 2 contains the formats of PP, OPL, OPLD, ULIB, and TEXT records. (The OPLC record format is identical to that of the OPL record.) The NOS Modify Reference Manual describes how to create OPL, OPLC, and OPLD records. The CYBER Loader Reference Manual contains the formats of ABS, CAP, OVL, PP, and REL records. The CYBER Loader Instant contains the PPU record format. Section 4 describes the creation of CCL procedures (PROC).

A user determines the record types contained in a file by issuing a CATALOG statement naming the file. VFYLIB lists the differences between the record types of two files.

CATALOG STATEMENT

The CATALOG control statement lists information about each record in a file assigned to the job.

NOTE

CATALOG produces unpredictable results when attempting to catalog an S, L, or F format tape. The user should use the COPY statement to convert the S, L, or F format tape to a mass storage file or to an I or SI binary format tape before attempting to catalog the file.

The control statement format is:

CATALOG(lfn, p_1, p_2, ..., p_n)

lfn — Name of the file to be cataloged.

p_i — May be any of the following.

N=0	Catalog until an empty file is encountered (two consecutive EOR marks).
N=n	Catalog n files; if N is omitted, N=1 is assumed.
N	Catalog until EOI is encountered.
L=fname	Specifies the name of the output file; if L is omitted, CATALOG assumes L=OUTPUT.
U	Lists records within a user library; if U is omitted, only the ULIB record within the user library is listed.
D	Suppresses the comments field; suppresses all page headings after the initial page heading for each file.
CS	Suppresses character set indicator (63 or 64) for OPL and OPLC records.
R	Rewinds lfn before and after cataloging; if R is omitted, lfn is not rewound.

The listing for each file of a multifile.file begins on a new page with a page heading for that file. If the D option has been specified, the page heading appears only once, at the beginning of the file. The information listed under each heading is as follows:

Heading	Information
REC	Record number (zero-length records and EOF marks are counted).
NAME	Record name [the contents of the name field from the second word of the prefix (77) table, if present; otherwise, the first 7 characters of the record].
TYPE	Record type (refer to Library Record Types in this section).
LENGTH	Record length in words (less prefix table length) printed as an octal number.
CKSUM	A checksum [a value used to verify that the contents of a record (excluding the prefix table) were copied correctly].
DATE	Record creation date (taken from the third word of the prefix table, if present).
COMMENTS	Additional information taken from the prefix table, if present; message terminates before COPYRIGHT comment. (This field is not shown when CATALOG is used in a time-sharing job.)

CATALOG lists additional information depending on the record type. Entry points are listed for REL and ABS records. The character set used, correction identifiers, and their YANK status (refer to the NOS Modify Reference Manual) are listed for OPL and OPLC records.

If the TEXT record name begins with CMRDC, CRMDECK, IPRDC, IPRDECK, LIBDC, or LIBDECK, CATALOG lists the entire record. If the TEXT record name begins with OVERLAY, CATALOG lists the first line in the record.

A ULIB record suppresses listing of the other records in the user library unless the U parameter is specified on the control statement.

When a zero-length record is encountered, the length since the last zero-length record is given. If an EOR does not precede an EOF or EOI within the cataloged file, the following message is output before the * EOF * or * EOI * line.

```
* EOR MISSING *
```

The ITEMIZE control statement (described in the Common Utilities Reference Manual) is similar to the CATALOG statement, but ITEMIZE recognizes additional record types.

Example:

Compilation of the FORTRAN Extended program A and its subroutines B and C wrote relocatable object code on file L. The following is a catalog of file L (refer to the heading definitions given earlier). The I/O file names listed in the FORTRAN Extended PROGRAM statement are flagged by # characters.

```
       CATALOG OF L                FILE    1                                   79/02/27. 16.13.27.       PAGE      1
REC    NAME      TYPE            LENGTH   CKSUM      DATE      COMMENTS
  1    A         REL                114    0706  79/02/27. 16.13.13   NOS 1.4  FTN     4.7485  666X  I        PROGRAM
        A
        INPUT#
        OUTPUT#
        TAPE6#
  2    B         REL                 25    4410  79/02/27. 16.13.13   NOS 1.4  FTN     4.7485  666X  I        SUBROUTINE
        B
  3    C         REL                 25    1450  79/02/27. 16.13.13   NOS 1.4  FTN     4.7485  666X  I        SUBROUTINE
        C

  4    * EOF *          SUM =       166
```

GTR STATEMENT

The GTR control statement appends records selected from one file to the end of another file. The records are selected according to directives specifying their type and name. Records can be accessed randomly (default if a directory exists) or sequentially. If specified, a random access directory is appended to the changed file. GTR cannot append records after the directory.

The control statement format is:

GTR(lfn_1, lfn_2, D, NR, S, NA)$directive_1$, $directive_2$, . . . , $directive_n$

NOTE

> The parameters must be in the order shown. GTR identifies its parameters by their position, not by keywords. An omitted parameter is denoted by consecutive commas. Blanks are illegal between the terminator (right parenthesis or period) and $directive_1$.

The parameters are defined as follows:

lfn_1 — File which is searched for the requested records; if lfn_1 is omitted, file OLD is assumed. lfn_1 is always rewound before the GTR operation.

lfn_2 — File on which the selected records are written; if lfn_2 is omitted, file LGO is assumed. GTR always positions lfn_2 at EOI before copying the selected records.

D — Writes a random access directory (OPLD) at the end of lfn_2. If D is specified, lfn_2 must be a mass storage file. GTR cannot append records after a directory.

This parameter has special meaning when the ULIB (user library) record type is specified in the directives.

If D is omitted, the first record of the user library, the user library directory (ULIB), is not copied to lfn_2; the last record (OPLD) is copied but not altered.

If D is specified, the first record of the user library (ULIB) is copied without alteration to lfn_2 along with the relocatable records and the old random access directory (OPLD). A new random access directory (OPLD) for the file is also added to lfn_2.

NR — If NR is specified, lfn_1 is not rewound after the operation; lfn_2 is not rewound before or after the operation. If lfn_1 has a directory, the directory is copied to lfn_2.

If NR is omitted, both files are rewound before and after the operation.

S — lfn_1 is searched sequentially; no attempt is made to read a directory.

NA — If specified, GTR does not search for an EXIT statement when an error occurs. It issues a dayfile message for the error and continues GTR processing at the next directive.

$directive_i$ — Specifies a record or group of records to be retrieved. One or more of the following formats can be used. Valid record types are listed under Library Record Types in this section. The default type is the last type specified on a directive, or if none specified, TEXT. The record name is the first 7 characters of the record, or if a prefix table is present, the contents of the name field in its second word.

Directive	Meaning
type/name	Record with the specified type and name.
name	Record with the specified name and the default type.
$type_1/name_1$-$type_2/name_2$	Group of records beginning with $name_1$ of $type_1$ and ending with $name_2$ of $type_2$.
$type_1/name_1$-$name_2$	Group of records beginning with $name_1$ of $type_1$ and ending with $name_2$ of $type_1$.
$name_1$-$name_2$	Group of records beginning with $name_1$ of the default type and ending with $name_2$ of the default type.
type/name-*	All records of the specified type beginning with the named record.
name-*	All records of the default type beginning with the named record.
type/*	All records of the specified type.

Directive	Meaning
*	All records of the default type.
0	A zero-length record is inserted.

GTR searches file lfn_1 for the records specified by the selection directives. If GTR cannot find a record specified by type and name, it issues the following dayfile message.

GTR ERRORS.

It also issues this message when the record specified is within a user library and when the GTR statement syntax is incorrect.

If lfn_2 is on tape, the selected records are copied from the current file position; if lfn_2 is on mass storage, the copy starts at the current EOI of the file.

Examples:

- GTR(SYSTEM,BIN,D)PP/*

  GTR copies all PP records from file SYSTEM to file BIN. It then builds a random access directory and writes it as the last record on BIN.

- GTR(OPL,NEW,,NR)OPLC/COMCARG,0,COMCCIO

  GTR retrieves common decks COMCARG and COMCCIO from file OPL. It then writes COMCARG, a zero-length record, and COMCCIO at the current position of file NEW. NEW is not rewound before the operation; OPL and NEW are not rewound following the operation.

- GTR(SYSTEM,SYSLIB,D)ULIB/SYSLIB

  GTR copies the user library SYSLIB from file SYSTEM to the end of file SYSLIB.

- GTR.REL/A

  GTR retrieves the relocatable record A from file OLD and copies it to file LGO.

LIBEDIT STATEMENT

LIBEDIT is a general-purpose utility that generates a file containing records copied from one or more other files (figure 1-14-2). LIBEDIT can build a random access directory for the new file. It recognizes the record types listed in Library Record Types in this section. LIBEDIT processes a user library as a single record.

LIBEDIT can edit a library according to directives requesting addition, deletion, or replacement of specified records from one or more replacement files. Replacement is the implicit mode of a LIBEDIT run. The user must explicitly identify records to be added and records that are not to be replaced (refer to the description of the NOREP directive).

LIBEDIT executes in two phases. During the first phase, it reads directives and replacement records. It groups directives by type and file and groups changes when several insertions take place relative to the same record.

During the second phase, LIBEDIT writes the new file. If LIBEDIT cannot process the specified combination of directives, and the D option (refer to the following control statement description) was not specified, LIBEDIT lists its interpretation of the conflicting directives, issues an error message, and aborts the job step. If the D option was specified, LIBEDIT continues processing the directives.

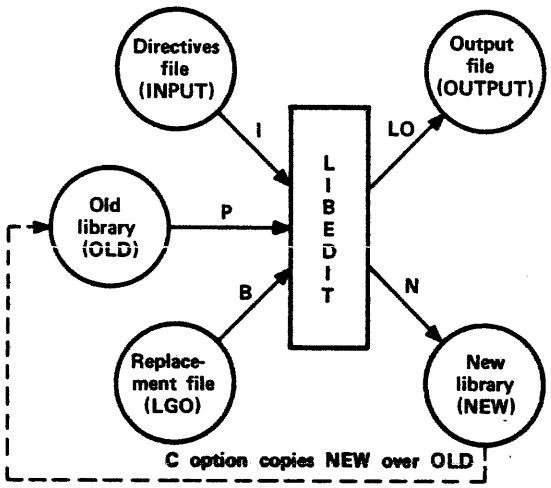


Figure 1-14-2. LIBEDIT Input and Output

CONTROL STATEMENT FORMAT

The following control statement calls LIBEDIT. Its parameters specify options and files used for the call as illustrated in figure 1-14-2.

LIBEDIT(p_1,p_2,...,p_n)

The option parameters p_i can be in any order. Each parameter cannot be specified more than once. lfn is a file name of 1 to 7 alphanumeric characters.

p_i	Meaning
P=lfn$_1$	Edit file lfn$_1$ (the old file).
P=0	No old file; new file created from replacement file(s).
P omitted	Edit file OLD.
N=lfn$_2$	Write new file on file lfn.
N=0	Illegal; error message issued.
N omitted	Write new file on file NEW.

p_i	Meaning
	NOTE The new file is evicted prior to processing (refer to the EVICT statement in section 7).
$I=lfn_3$	Take directives from the next record of lfn_3.
I=0	No directive input.
I omitted	Take directives from file INPUT.
$B=lfn_4$	Use records from file lfn_4 for insertions and replacements.
B=0	No replacement file used (unless specified by a FILE directive).
B omitted	Use file LGO as a replacement file.
$LO=lfn_5$	List output on file lfn_5.
LO=0	List no output.
LO omitted	List output on file OUTPUT.
L=1	List directives, modifications, and errors on output file.
L=0	List only errors.
L omitted	Same as L=1.
V	Call VFYLIB after LIBEDIT processing.
V omitted	Do not call VFYLIB.
R	Do not rewind files after processing.
R omitted	Rewind old and new files after processing.
C	Copy the new file over the old file after processing.
C omitted	Do not copy the new file over the old file.
D	Ignore errors and continue.
D omitted	Do not ignore errors; abort job step.

LIBEDIT DIRECTIVES

The user can specify directives to control LIBEDIT processing. These directives can be in a record on file INPUT or on the file specified by the control statement I parameter.

Directives are not required. If I=0 is specified, LIBEDIT compares the name and type of each record on the old file with those of the records on the replacement file (specified by the B parameter). If a record with the same name and type appears on the replacement file, LIBEDIT writes that record on the new file and skips the record on the old file; otherwise, it copies the record from the old file to the new file. If I=0 and B=0 are specified, LIBEDIT copies the old file to the new file until it encounters an EOF mark or an OPLD directory on the old file.

LIBEDIT recognizes the following directives.

Directive	Function
*ADD	Inserts records before a zero-length record within the file.
*BEFORE or *B	Inserts record before the named record.
*BUILD	Builds a directory at the end of the new file.
*COMMENT	Adds a comment to the prefix table.
*COPY	Copies the new file to the old at the end of editing.
*DATE	Adds the date and a comment to the prefix table.
*DELETE or *D	Does not copy specified records to the new file.
*FILE	Declares a file to be a replacement file.
*IGNORE	Ignores records when reading the replacement file.
*INSERT or *I or *AFTER or *A	Copies record from the replacement file after copying the specified old file record.
*NOREP	Does not automatically replace old file records with records from the specified file.
*RENAME	Rename record.
*REPLACE	Replaces the named records from the old file with records from the replacement file.
*REWIND	Names file to be rewound before and after editing.
*TYPE or *NAME	Sets default record type.

Directive Syntax

A directive begins with an asterisk in column 1 followed immediately by the directive identifier. The directive identifier is delimited by a comma and/or one or more spaces. If a directive does not begin with an asterisk and a directive identifier, LIBEDIT assumes that the operation is a continuation of the previous directive operation. If an asterisk and directive identifier do not begin the first line of the directives record, LIBEDIT prefixes the following to the first line.

```
*BEFORE *.
```

Parameters are delimited by a blank, an end-of-line, or a comma. A hyphen (-) indicates a record group. Record group identifiers (gid entries) cannot be split between lines. For example, the lines

```
*B,OVL/P1,OVL/P2,OVL/P
3
```

do not constitute a valid directive. The last entry would not be processed as OVL/P3. On the other hand, the lines

```
*B,OVL/P1,OVL/P2
OVL/P3
```

do constitute a valid directive and would be processed as:

```
*B,OVL/P1,OVL/P2,OVL/P3
```

Parameters common to many directives are the reference record identifier (rid) and the record group identifier (gid). Valid record types for these parameters are listed in Library Record Types in this section. The default type is the last type specified in a directive; if none are specified, TEXT is the default. The record name is the first 7 characters of the record, or if a prefix table is present, the name in its second word. The first character of a record name specified in a directive must not be an asterisk.

rid Reference record identifier specifying the reference point for the requested change. It can have the following formats.

type/name	Reference record has the specified type and name.
name	Reference record has the specified name and is of the default type.
*	Reference point is an end-of-file mark (used with *BEFORE directive only).

gid Record group identifier indicating a record or group of records to be inserted, deleted, or replaced. It can have the following formats.

type/name	Record with the specified type and name.
name	Record with the specified name of the default type.
$type_1/name_1$-$type_2/name_2$	Group of records beginning with $name_1$ of $type_1$ and ending with $name_2$ of $type_2$.
$type_1/name_1$-$name_2$	Group of records beginning with $name_1$ of $type_1$ and ending with $name_2$ of $type_1$.
$name_1$-$name_2$	Group of records beginning with $name_1$ of the default type and ending with $name_2$ of the default type.
type/name-*	All records of the specified type beginning with the named record.
name-*	All records of the default type beginning with the named record.
type/*	All records of the specified type.
*	All records of the default type.
0	A zero-length record is inserted.

ADD

The ADD directive inserts records before a zero-length record. A CATALOG listing of the old file numbers each group of records ending with a zero-length record (called a library on the listing). This number on the ADD directive identifies the record group.

NOTE

Adding a zero-length record does not change the directory.

The directive format is:

*ADD LIBn, $gid_1, gid_2, \ldots gid_n$

LIBn — Specifies the record group to which the records are appended. Values for n are 1 to 63 and can be determined from a CATALOG listing of the old file.

gid_1 — Identifies the records or groups of records from the current replacement file that are to be inserted before the zero-length record.

Example:

The following is a CATALOG listing of file Q.

```
        CATALOG OF Q              FILE     1
  REC   NAME      TYPE          LENGTH    CKSUM      DATE

   1    REC1      TEXT               1     5302
   2    REC2      TEXT               1     5304
   3    (00)          SUM =          2   LIBRARY =          1

   4    REC4      TEXT               1     5310

        * EOI *       SUM =          3
```

The following output results when a record was added to file Q, producing file Y.

```
 LIBEDIT DIRECTIVE CARDS.                              78/11/13. 15.12.51.        PAGE     1

     *ADD LIB1,REC3
 RECORDS WRITTEN ON FILE Y                             78/11/13. 15.12.51.        PAGE     2
     RECORD     TYPE       FILE        DATE        COMMENT
     REC1       TEXT       Q
     REC2       TEXT       Q
INSERTED REC3   TEXT       X
     00                    Q

     REC4       TEXT       Q
     **EOF**               Q
```

BEFORE

A BEFORE directive inserts records or groups of records before a specified reference record on the old file. An old file record with the same name and type as an inserted record is not copied to the new file.

The directive formats are:

*BEFORE rid,gid_1,gid_2,...,gid_n

or

*B rid,gid_1,gid_2,...,gid_n

rid	Names the old file record before which the specified replacement file records are to be inserted.
gid_1	Identifies records or groups of records from the current replacement file that are to be inserted before the reference record (rid).

If the first line of the LIBEDIT directives record does not begin with an asterisk and directive name, LIBEDIT assumes that the line is the gid parameters following a *BEFORE *, directive.

BUILD

A BUILD directive constructs and appends a random access directory to the new file. The directory is in Modify format (an OPLD record). If the old file has an OPLD directory, LIBEDIT constructs a directory for the new file with or without a BUILD directive. BUILD can also be used to change the directory name.

The directive format is:

*BUILD dname

dname	1- to 7-alphanumeric character name for the directory record. No default.

COMMENT

The COMMENT directive adds a comment to the prefix (77) table of a record written on the new file.

The directive format is:

*COMMENT rid comment

rid	Name of a record to be written on the new file.
comment	A string of up to 40 characters that appears in the comment field of the prefix table. Additional characters are truncated.

COPY

The COPY directive directs LIBEDIT to copy the new file over the old file after it has processed all directives.

The directive format is:

*COPY lfn_2,lfn_1

lfn_2	Name of new file to be copied over old file.
lfn_1	Name of old file to be overwritten.

It performs the same function as the C parameter on the LIBEDIT control statement. If files other than the old file and the new file (as specified on the LIBEDIT statement) are named, the directive is ignored.

DATE

The DATE directive adds the current date and the specified comment to the prefix (77) table of a record written on the new file.

The directive format is:

*DATE rid comment

rid	Record to be written on the new file.
comment	A string of up to 40 characters to be written in the comment field of the prefix table. Additional characters are truncated.

DELETE

The DELETE directive suppresses copying of the specified records from the old file to the new file.

The directive formats are:

*DELETE gid_1,gid_2,...,gid_n

or

*D gid_1,gid_2,...,gid_n

gid_i	Identifies records or groups of records that are not to be copied from the old file to the new file.

Example:

*DELETE OVL/LAD-REL/RUN

This directive requests LIBEDIT not to copy the sequence of records starting with overlay LAD through relocatable CPU program RUN.

FILE

The FILE directive names a file assigned to the job that contains replacement records. LIBEDIT directives following the FILE directive refer to records on the declared replacement file.

The directive format is:

*FILE lfn

lfn	1- to 7-character name of a replacement file. If lfn is an asterisk (*), LIBEDIT uses the replacement file specified by the LIBEDIT control statement. If the B parameter was omitted from the control statement, LGO is used.

IGNORE

The IGNORE directive requests LIBEDIT to ignore a record or group of records on the current replacement file.

The directive format is:

*IGNORE $gid_1, gid_2, \ldots, gid_n$

gid_i	Identifies records or groups of records on the replacement file that are to be ignored.

Example:

```
*FILE ALPHA
*IGNORE C-*
```

LIBEDIT ignores the sequence of records on file ALPHA starting with record C of the default type and including all records of the default type from C to the EOF mark.

INSERT OR AFTER

The INSERT or AFTER directive requests LIBEDIT to copy the specified records or groups of records from the current replacement file after it has copied the specified old file record onto the new file. Any record on the old file that has the same name and type as an inserted record is not copied to the new file.

The formats for the directives are:

*INSERT $rid, gid_1, gid_2, \ldots, gid_n$

or

*I $rid, gid_1, gid_2, \ldots, gid_n$

*AFTER $rid, gid_1, gid_2, \ldots, gid_n$

or

*A $rid, gid_1, gid_2, \ldots, gid_n$

Example:

*INSERT OPL/K,TEXT/L

This directive requests LIBEDIT to copy the replacement file text record L to the new file after it has copied the old file OPL record K.

NOREP

The NOREP directive declares the specified files to be no-replace files. A no-replace file is a replacement file whose records do not automatically replace old file records having the same name and type. The user selects records to be written on the new file from no-replace files by specifying the file on a FILE directive and then naming the records on *AFTER, *BEFORE, *INSERT, and *REPLACE directives.

The directive format is:

*NOREP $lfn_1,lfn_2,\ldots,lfn_n$

RENAME

The RENAME directive assigns a new name to a record written on the new file. If the renamed record is referenced by another directive in the directive record, the old name should be used. A RENAME is not allowed on a PROC record.

The directive format is:

*RENAME rid,name

rid	Name of the replacement file record or old file record to be renamed.
name	1- to 7-alphanumeric character new name of the record.

REPLACE

The REPLACE directive requests LIBEDIT to replace the old file records having the specified names and types with the replacement file records having matching names and types. This directive is used when the current replacement file has been declared a no-replace file (refer to the NOREP directive description). If the replacement file is not a no-replace file, LIBEDIT performs the replace operation automatically.

The directive format is:

*REPLACE $gid_1,gid_2,\ldots,gid_n$

gid_i	Specifies records or groups of records that appear on both the old file and the current replacement file.

Example:

The old file contains text records A, B, C, and D; the replacement file RF also contains text records named A, B, C, and D. Either of the following directive sequences writes records A and B from the old file and records C and D from file RF onto the new file.

Sequence 1	Sequence 2
*FILE RF	*FILE RF
*NOREP RF	*IGNORE A-B
*REPLACE C-D	

REWIND

The REWIND directive tells LIBEDIT to rewind the specified file before and after processing.

The directive format is:

```
*REWIND lfn
```

lfn	Name of the file to be rewound.

TYPE OR NAME

A TYPE or NAME directive sets the default record type.

The directive formats are:

```
*TYPE type
*NAME type
```

type	Specifies default record types. Valid record types are listed in Library Record Types in this section.

The default record type can also be set by an explicit record type specification within a directive. In either case, the default record type setting remains in effect until another record type is explicitly named. If a default record type is not declared in the directive sequence, the default is TEXT. For example, the following two directive sequences are equivalent.

Sequence 1	Sequence 2
*TYPE REL	*INSERT REL/X,Y
*INSERT X,Y	*DELETE FILE1-FILE4
*DELETE FILE1-FILE4	

LIBEDIT OUTPUT

LIBEDIT interprets all directives in the directive record before beginning directive processing. If one or more errors are found, LIBEDIT issues the dayfile message

DIRECTIVE CARD ERROR.

and aborts the job step (unless the D parameter is specified on the control statement). The following LIBEDIT output shows the results of a directive syntax error (the FILE directive is not followed by a space or comma).

```
     LIBEDIT DIRECTIVE CARDS.                          78/11/15. 10.36.47.          PAGE     1
*ERROR*  *FILERF1
```

Directives which cannot be executed are listed as LIBEDIT interpreted them. The following LIBEDIT run called for a replacement file not assigned to the job.

```
     LIBEDIT DIRECTIVE CARDS.                          78/11/15. 10.37.39.          PAGE     1
         *FILE RF1
         *B *,X
*ERROR*  DIRECTIVE CARD CAN NOT BE FOLLOWED.
         *FILE     RF1
         *BEFORE   TEXT/*,TEXT/X
```

Nonfatal errors are listed in an error directory following the listing of records written to the new file. The RECORDS NOT REPLACED error shown in the following example could be corrected by including an *IGNORE directive naming the records not to be replaced.

```
     LIBEDIT DIRECTIVE CARDS.                          79/02/28. 08.37.43.          PAGE     1
         *FILE RF2
         *B *,REC1
     RECORDS WRITTEN ON FILE NEW                       79/02/28. 08.37.43.          PAGE     2
         RECORD    TYPE      FILE       DATE      COMMENT
INSERTED REC1      TEXT      RF2
         **EOF**             OLD

     ERROR DIRECTORY - RECORDS NOT REPLACED.           79/02/28. 08.37.43.          PAGE     3
         RECORD    TYPE      FILE
         REC3      TEXT      RF2
         REC4      TEXT      RF2
```

LIBGEN STATEMENT

The LIBGEN control statement generates a user library of routines for use with CYBER Loader. The control statement format is:

LIBGEN(p_1,p_2,...,p_n)

Any of the following parameters may be specified in any order (only one instance of each).

p_i	Meaning
F=lfn_1	Source file containing the relocatable (REL) and/or capsule (CAP) records for the user library. Other record types are ignored.
F or F omitted	LGO is the source file.
P=lfn_2	File on which the user library is to be written.
P or P omitted	User library is to be written on file ULIB.
N=name	Name of the user library generated; name entered in ULIB and OPLD records.
N or N omitted	User library name specified by P parameter.
NX=n	If n is not zero, no cross-references are included in the ULIB directory. If n is zero, cross-references are included.
NX or NX omitted	LIBGEN assumes NX=0 (cross-references are included in the directory).

If the F and P parameters specify the same file, LIBGEN issues a dayfile message and does not generate a user library.

Figure 1-14-3 illustrates the structure of a user library. To generate a user library, LIBGEN rewinds and scans the source file, building a directory of all entry points, program names, and external references in the relocatable and capsule records in the file. LIBGEN then copies the source file to the user library file adding the ULIB and OPLD records.

Unless the NX parameter specifies otherwise, the ULIB directory contains the external reference/entry point linkage between routines in the user library. When CYBER Loader loads a routine from the user library, it loads (at the same time) all user library routines referenced by the requested routine. All externals for user library routines are satisfied from the user library, if possible. If desired, the user can request with the NX parameter that the ULIB directory contain no cross-linking of records. In that case, when a routine from the user library is requested, only that routine is loaded.

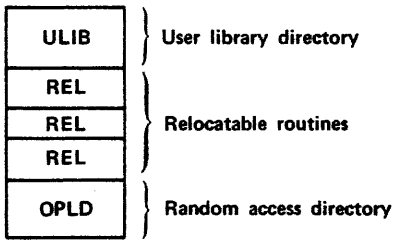


Figure 1-14-3. User Library Structure

Example 1:

File RELB contains relocatable routines that are used for execution of several applications. To enable loading of these routines as needed during execution of an application program, the user generates a user library using the following control statement.

```
LIBGEN(F=RELB,P=MYLIB,N=APPLIB)
```

This creates user library APPLIB on file MYLIB. The following loader sequence allows use of the APPLIB routines during execution of a compiled FORTRAN Extended program on file LGO.

```
LDSET(LIB=MYLIB)
LOAD(LGO)
EXECUTE.
```

The program is loaded and executed with externals satisfied first from user library MYLIB and then from the system default library SYSLIB. Refer to the CYBER Loader Reference Manual for more information on library search procedures.

Example 2:

If a routine has no external references, no entry is made in the ULIB directory. To load this routine, the user must include the loader statement LDSET(USEP=pname) in a loader sequence.

Suppose a FORTRAN Extended program contains a BLOCK DATA subroutine without external references to any of its entry points. The user has not named the block, and it has the default name BLKDAT. To load this routine, the user must include the following control statement in the loader sequence.

```
LDSET(USEP=$BLKDAT.$)
```

VFYLIB STATEMENT

The VFYLIB control statement rewinds two files, compares their record sequence, and lists the differences. VFYLIB lists changes in residence (between record groups separated by zero-length records), replacements, deletions, and insertions. A record is defined as being replaced when its name and type remain the same, but its contents differ. VFYLIB does not compare prefix (77) table information such as last modification date and last assembly date.

The control statement format is:

VFYLIB(lfn_1,lfn_2,lfn_3,NR)

lfn_1	Name of the first file; if this parameter is omitted, VFYLIB assumes file OLD.
lfn_2	Name of the second file; if this parameter is omitted, VFYLIB assumes files NEW.
lfn_3	Name of the output file; if this parameter is omitted, VFYLIB assumes file OUTPUT.
NR	If specified, lfn_1 and lfn_2 are not rewound after verification.

Example:

The following are CATALOG listings of file OLD and file NEW.

REC	CATALOG OF OLD NAME	TYPE	FILE LENGTH	1 CKSUM	DATE
1	A	REL	25	7547	79/02/23.
2	(00)	SUM =	25	LIBRARY =	1
3	B	REL	25	4410	79/02/28.
4	C	REL	25	1450	79/02/28.
5	* EOF *	SUM =	77		

REC	CATALOG OF NEW NAME	TYPE	FILE LENGTH	1 CKSUM	DATE
1	A	REL	30	4122	79/02/28.
2	B	REL	25	4410	79/02/28.
3	(00)	SUM =	55	LIBRARY =	1
4	D	TEXT	1	1000	
5	* EOF *	SUM =	56		

The control statement, VFYLIB., produces the following listing.

```
VFYLIB.      OLD FILE = OLD      NEW FILE = NEW      79/02/23. 08.58.32.          PAGE     1
     RECORD     TYPE      ULIB          LIB    DATE       COMMENT

RECORDS REPLACED.
     A          REL                      1    79/02/23. 08.56.04   NOS 1.4  FTN    4.7435  666X  I       SUBROUTINEOPT=1
CHANGES IN RESIDENCE.
     B          REL                      1    79/02/23. 08.40.14   NOS 1.4  FTN    4.7435  665X  I       SUBROUTINEOPT=1
DELETED PROGRAMS.
     C          REL                      2    79/02/23. 08.40.14   NOS 1.4  FTN    4.7435  665X  I       SUBROUTINEOPT=1
INSERTED PROGRAMS.
     D          TEXT                     2
```

LIBRARY PROCESSING EXAMPLES

The following examples illustrate the use of CATALOG, GTR, LIBEDIT, and LIBGEN control statements. To duplicate the examples, the user should execute the jobs in sequence.

Example 1:

The following job builds a program library from a replacement file of relocatable binary (REL) records.

```
LIBTES1.
USER(EFD25,PW)
CHARGE(16,13N122)
FTN(L=0)
DEFINE(TESTLIB)
CATALOG(LGO,R)
LIBEDIT(P=0,N=TESTLIB)
CATALOG(TESTLIB,R)
-EOR-
    SUBROUTINE A
    STOP
    END
    SUBROUTINE D
    STOP
    END
    SUBROUTINE C
    STOP
    END
    SUBROUTINE B
    STOP
    END
/EOR
*BUILD LIBRARY
*B,*,REL/A,B,C,D
/EOF
```

The FORTRAN Extended compilation produces relocatable binaries on the default file LGO. The DEFINE statement creates a direct access permanent file TESTLIB on which the new program library is written. The first CATALOG statement lists the LGO file as follows:

```
     CATALOG OF LGO            FILE     1                          79/03/01. 08.17.27.      PAGE       1
REC  NAME         TYPE       LENGTH   CKSUM     DATE       COMMENTS
 *
  1  A            REL           25     7547  79/03/01. 08.16.29  NOS 1.4  FTN    4.7485  666X  I       SUBROUTINE
      A
  2  D            REL           25     6705  79/03/01. 08.16.29  NOS 1.4  FTN    4.7485  666X  I       SUBROUTINE
      D
  3  C            REL           25     1450  79/03/01. 08.16.29  NOS 1.4  FTN    4.7485  666X  I       SUBROUTINE
      C
  4  B            REL           25     4410  79/03/01. 08.16.29  NOS 1.4  FTN    4.7485  666X  I       SUBROUTINE
      B

  5  * EOF *      SUM =        124
```

The P=0 in the LIBEDIT statement indicates that no old program library exists. The N parameter indicates that the new program library is written on file TESTLIB. The replacement file is the default LGO. The directives are on the default INPUT file.

LIBEDIT reads the binaries from LGO and the directives from INPUT. On the basis of the directive specifications, the binaries are inserted before the end-of-file on file TESTLIB in the order specified in the directives (A, B, C, D). The directory record created is given the name LIBRARY as a result of the *BUILD directive. It is written before the end-of-file on the new program library TESTLIB.

The directives are written to OUTPUT. The records on file TESTLIB are listed on the next page of OUTPUT. The following listing consists of these two pages.

```
 LIBEDIT DIRECTIVE CARDS.                              79/03/01. 08.20.52.          PAGE     1

     *BUILD LIBRARY
     *B, *,REL/A,B,C,D
```

```
     RECORDS WRITTEN ON FILE TESTLIB                   79/03/01. 08.20.52.          PAGE     2
         RECORD    TYPE      FILE      DATE      COMMENT
INSERTED A         REL       LGO       79/03/01. 08.15.29   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINEOPT=1
INSERTED B         REL       LGO       79/03/01. 08.15.29   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINEOPT=1
INSERTED C         REL       LGO       79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINEOPT=1
INSERTED D         REL       LGO       79/03/01. 08.15.29   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINEOPT=1
ADDED    LIBRARY   OPLD      *****     79/03/01.
         **EOF**
```

The second CATALOG statement produces the following listing of information about the records on TESTLIB.

```
     CATALOG OF TESTLIB          FILE     1                    79/03/01. 08.21.46.          PAGE     1
REC  NAME      TYPE          LENGTH    CKSUM     DATE      COMMENTS
  1  A         REL             25      7547  79/03/01. 08.15.29   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINE
       A
  2  B         REL             25      4410  79/03/01. 08.15.29   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINE
       B
  3  C         REL             25      1450  79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINE
       C
  4  D         REL             25      6705  79/03/01. 08.15.29   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINE
       D
  5  LIBRARY   OPLD            13      2073  79/03/01.
  6  * EOF *       SUM =      137
```

Example 2:

This job builds a new program library from an old program library by inserting new relocatable routines into and deleting routines from the old program library created in example 1 (TESTLIB).

```
LIBTES2.
USER(EFD2S,PW)
CHARGE(16,13N122)
FTN(L=0)
ATTACH(OLD=TESTLIB)
DEFINE(NEW=TES2LIB)
LIBEDIT.
CATALOG(NEW,R)
-EOR-
          SUBROUTINE BONE
          STOP
          END
          SUBROUTINE D
          STOP
          END
          SUBROUTINE NEWC
          STOP
          END
/EOR
*TYPE REL
*I,B,BONE
*I,C,NEWC
*D,C
/EOF
```

Three relocatable binaries (BONE, D, and NEWC) are produced via a FORTRAN Extended compilation. The old program library (TESTLIB) is attached in read mode and referenced as OLD. A direct access file (TES2LIB) is created for the new program library. This file is referenced as NEW.

LIBEDIT reads the binaries from the replacement file LGO and the input directives from file INPUT. It writes the modified old program library (OLD) to the new program library (NEW). BONE and NEWC are inserted after records B and C, respectively, and record C is deleted. Record D, which already existed on the old program library, is replaced by record D from the replacement file LGO. The following action is taken on file NEW.

```
     LIBEDIT DIRECTIVE CARDS.                                  79/03/01. 08.25.52.          PAGE     1

         *TYPE REL
         *I,B,BONE
         *I,C,NEWC
         *D,C
     RECORDS WRITTEN ON FILE NEW                               79/03/01. 08.25.52.          PAGE     2
         RECORD     TYPE       FILE       DATE       COMMENT
         A          REL        OLD        79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINEOPT=1
         B          REL        OLD        79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINEOPT=1
INSERTED BONE       REL        LGO        79/03/01. 08.25.07   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINEOPT=1
DELETED-(C)         REL        OLD
INSERTED NEWC       REL        LGO        79/03/01. 08.25.07   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINEOPT=1
REPLACED D          REL        LGO        79/03/01. 08.25.07   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINEOPT=1
ADDED    LIBRARY    OPLD       *****      79/03/01.
         **EOF**               OLD
```

The CATALOG shows the following contents of the new program library.

```
     CATALOG OF NEW          FILE    1                               79/03/01. 08.26.48.       PAGE       1
REC  NAME       TYPE       LENGTH   CKSUM     DATE       COMMENTS

  1  A          REL            25    7547  79/03/01. 08.16.29  NOS 1.4  FTN    4.7485  666X  I          SUBROUTINE
      A
  2  B          REL            25    4410  79/03/01. 08.16.29  NOS 1.4  FTN    4.7485  666X  I          SUBROUTINE
      B
  3  BONE       REL            25    1103  79/03/01. 08.25.07  NOS 1.4  FTN    4.7435  666X  I          SUBROUTINE
      BONE
  4  NEWC       REL            25    0371  79/03/01. 08.25.07  NOS 1.4  FTN    4.7485  666X  I          SUBROUTINE
      NEWC
  5  D          REL            25    6705  79/03/01. 08.25.07  NOS 1.4  FTN    4.7485  666X  I          SUBROUTINE
      D
  6  LIBRARY    OPLD           15    1312  79/03/01.
  7  * EOF *        SUM =     166
```

Example 3:

This job uses LIBGEN to generate a user library file from the program library file TES2LIB created in example 2.

```
LIBTES3.
USER(EFD25)
CHARGE(16,13N122)
ATTACH(TES2LIB)
DEFINE(LIBLOAD)
LIBGEN(F=TES2LIB,P=LIBLOAD,N=LOADLIB)
CATALOG(LIBLOAD,R,U)
-EOF-
```

The program library TES2LIB is attached to the job. A direct access file LIBLOAD is defined for writing the user library file.

LIBGEN scans TES2LIB and builds a ULIB directory of entry points, program names, and external references for relocatable (REL) records in the file. ULIB is copied to the file LIBLOAD, followed by the records from TES2LIB. A file index of addresses for each record in the file is added as the last record of LIBLOAD. LOADLIB is the name of the ULIB and OPLD records.

The CATALOG of the user library file LIBLOAD shows the following content.

```
     CATALOG OF LIBLOAD      FILE    1                               79/03/01. 08.29.02.       PAGE       1
REC  NAME       TYPE       LENGTH   CKSUM     DATE       COMMENTS

  1  LOADLIB    ULIB           13    4267  79/03/01.
  2  A          REL            25    7547  79/03/01. 08.16.29  NOS 1.4  FTN    4.7485  666X  I          SUBROUTINE
      A
  3  B          REL            25    4410  79/03/01. 08.16.29  NOS 1.4  FTN    4.7485  666X  I          SUBROUTINE
      B
  4  BONE       REL            25    1103  79/03/01. 08.25.07  NOS 1.4  FTN    4.7435  666X  I          SUBROUTINE
      BONE
  5  NEWC       REL            25    0371  79/03/01. 08.25.07  NOS 1.4  FTN    4.7435  666X  I          SUBROUTINE
      NEWC
  6  D          REL            25    6705  79/03/01. 08.25.07  NOS 1.4  FTN    4.7485  666X  I          SUBROUTINE
      D
  7  LOADLIB    OPLD           15    6303  79/03/01.
  8  * EOF *        SUM =     201
```

Example 4:

This job illustrates a method for deleting records from a user library. GTR removes the relocatable records (REL) from the user library, LIBEDIT makes the desired changes, and LIBGEN generates a new user library.

```
LIBTES4.
USER(EFD25,PW)
CHARGE(16,13N122)
ATTACH(LIBLOAD/M=W)
GTR(LIBLOAD,OLD)REL/*
LIBEDIT.
LIBGEN(F=NEW,P=LIBLOAD,N=LOADLIB)
CATALOG(LIBLOAD,R,U)
-EOR-
*D,REL/NEWC
-EOF-
```

The user library generated in example 4 (LIBLOAD) is attached to the job's control point.

Because LIBEDIT handles a user library as a single record, the GTR statement must be used to extract the relocatable records from LIBLOAD and write them on the file OLD. (This control statement terminates after OLD; the REL/* is a directive specifying all relocatable records.)

LIBEDIT references the program library OLD and the directive record, deletes NEWC, and writes this modified file on the default NEW. The following is a listing of NEW.

```
   LIBEDIT DIRECTIVE CARDS.                                   79/03/01. 08.30.06.         PAGE      1

       *D,REL/NEWC
     ----------------------------------------------------------------------------------------------

   RECORDS WRITTEN ON FILE NEW                                79/03/01. 08.30.06.         PAGE      2
       RECORD      TYPE      FILE      DATE        COMMENT
       A           REL       OLD      79/03/01. 08.15.29  NOS 1.4  FTN    4.7485  666X  I        SUBROUTINEOPT=1
       B           REL       OLD      79/03/01. 08.16.29  NOS 1.4  FTN    4.7435  666X  I        SUBROUTINEOPT=1
       BONE        REL       OLD      79/03/01. 08.25.07  NOS 1.4  FTN    4.7485  666X  I        SUBROUTINEOPT=1
DELETED-(NEWC)     REL       OLD
       D           REL       OLD      79/03/01. 08.25.07  NOS 1.4  FTN    4.7485  666X  I        SUBROUTINEOPT=1
       **EOF**               OLD
```

LIBGEN generates a new user library on the file LIBLOAD. It uses NEW as the source and names the new user library LIBLOAD.

The user library is cataloged, showing the following contents.

```
       CATALOG OF LIBLOAD      FILE     1                                  79/03/01. 08.30.54.       PAGE       1
REC    NAME       TYPE        LENGTH  CKSUM    DATE       COMMENTS

  1    LOADLIB    ULIB          11     1055  79/03/01.
  2    A          REL           25     7547  79/03/01. 08.15.29   NOS 1.4  FTN    4.7435  666X  I        SUBROUTINE
         A
  3    B          REL           25     4410  79/03/01. 08.15.29   NOS 1.4  FTN    4.7435  666X  I        SUBROUTINE
         B
  4    BONE       REL           25     1103  79/03/01. 08.25.07   NOS 1.4  FTN    4.7435  666X  I        SUBROUTINE
         BONE
  5    D          REL           25     6705  79/03/01. 08.25.07   NOS 1.4  FTN    4.7435  666X  I        SUBROUTINE
         D
  6    LOADLIB    OPLD          13     0414  79/03/01.
```

CHARACTER SETS A

A character set is composed of graphic and/or control characters. A code set is a set of codes used to represent each character within a character set.

A graphic character may be displayed at a terminal or printed by a line printer. Examples are the characters A through Z and the digits 1 through 9. A control character initiates, modifies, or stops a control operation. An example is the backspace character that moves the terminal carriage or cursor back one space. Although a control character is not a graphic character, a terminal may produce a graphic representation when it receives a control character.

All references within this manual to the ASCII character set or the ASCII code set refer to the character set and code set defined in the American National Standard Code for Information Interchange (ASCII, ANSI Standard X3.4-1977). References in this manual to the ASCII character set do not necessarily refer to the ASCII code set.

NOS supports the following character sets.

- CDC graphic 64- (or 63-) character set
- ASCII 128-character set
- ASCII graphic 64- (or 63-) character set
- ASCII graphic 95-character set

Each installation selects either the 64-character set or the 63-character set. The differences between the two are described in Character Set Anomalies in this appendix. Any reference in this appendix to the 64-character set implies either the 63- or 64-character set unless otherwise stated.

NOS supports the following code sets.

- Display code
- 6/12 display code
- 12-bit ASCII code

Display code is a set of 6-bit codes from 00_8 to 77_8.

The 6/12 display code is a combination of 6-bit codes and 12-bit codes. The 6-bit codes are 00_8 through 77_8, excluding 74_8 and 76_8. (Refer to Character Set Anormalies for the interpretation of the 00_8 and 63_8 codes.) The 12-bit codes begin with either 74_8 or 76_8 and are followed by a 6-bit code. Thus, 74_8 and 76_8 are considered escape codes and are never used as 6-bit codes within the 6/12 display code set. The 12-bit codes are 7401_8, 7402_8, 7404_8, 7407_8, and 7601_8 through 7677_8. All other 12-bit codes ($74xx_8$ and 7600_8) are undefined.

The 12-bit ASCII code is the ASCII 7-bit code (as defined by ANSI Standard X3.4-1977) right-justified in a 12-bit byte. Assuming that the bits are numbered from the right starting with 0, bits 0 through 6 contain the ASCII code, bits 7 through 10 contain zeros, and bit 11 distinguishes the 12-bit ASCII 0000_8 code from the end-of-line byte. The 12-bit codes are 0001_8 through 0177_8 and 4000_8.

CHARACTER SET ANOMALIES

NOS interprets two codes differently when the installation selects the 63-character set rather than the 64-character set. In tables 1-A-1, 1-A-2, and 1-A-3, the codes for the colon and percent graphic characters in the 64-character set are unshaded; the codes for the colon and percent graphic characters in the 63-character set are shaded.

If an installation uses the 63-character set, the colon graphic character is always represented by a 63_8 code. However, if the installation uses the 64-character set, output of 6/12 display codes 7404_8 or 00_8 produces a colon. In ASCII time-sharing mode, colon can be input only as a 7404_8 6/12 display code.

When using either the 63- or 64-character set, the use of undefined 6/12 display codes in output files produces unpredictable results and should be avoided.

Also, two 00_8 codes may be confused with an end-of-line byte and should be avoided (refer to appendix F for further explanation).

CHARACTER SET TABLES

This appendix contains character set tables for time-sharing users, batch users, and magnetic tape users. Table 1-A-1 is for time-sharing users, and table 1-A-2 is for batch users. Table 1-A-3 is a conversion table used to cross-reference 12-bit ASCII codes and 6/12 display codes and to convert ASCII codes from octal to hexadecimal.

Tables 1-A-4 and 1-A-5 list the magnetic tape codes and their display code equivalents.

The character set tables are designed so that the user can find the character represented by a code (such as in a dump) or find the code that represents a character. To find the character represented by a code, the user looks up the code in the column listing the appropriate code set and then finds the character on that line in the column listing the appropriate character set. To find the code that represents a character, he first looks up the character and then finds the code on the same line in the appropriate column.

TIME-SHARING USERS

Table 1-A-1 shows the character sets and code sets available to an ASCII code terminal user. When in NORMAL time-sharing mode (specified by the NORMAL time-sharing command), NOS displays the ASCII graphic 64-character set and interprets all input and output as display code. When in ASCII time-sharing mode (specified by the ASCII time-sharing command), NOS displays the ASCII 128-character set and interprets all input and output as 6/12 display code.

The time-sharing user can convert a 6/12 display code file to a 12-bit ASCII code file using the FCOPY control statement (section 7). The resulting 12-bit ASCII file can be routed to a line printer (refer to the ROUTE statement in section 7) but cannot be output at a time-sharing terminal.

To determine the octal or hexadecimal ASCII code for a character, refer to table 1-A-3. (Certain terminal definition commands require specification of an ASCII code.)

BATCH USERS

Table 1-A-2 lists the CDC graphic 64-character set, the ASCII graphic 64-character set, and the ASCII graphic 95-character sets. It also lists the code sets and card punch codes (O26 and O29) that represent the characters.

The 64-character sets use display code as their code set; the 95-character set uses 12-bit ASCII code. The 95-character set is composed of all the characters in the ASCII 128-character set that can be printed at a line printer (refer to Line Printer Usage). Only 12-bit ASCII code files can be printed using the ASCII graphic 95-character set. To print a 6/12 display code file (usually created in time-sharing ASCII mode), the user must convert the file to 12-bit ASCII code. To do this, he issues the FCOPY control statement (section 7). The 95-character set is represented by 12-bit ASCII codes 0040_8 through 0176_8.

LINE PRINTER USAGE

The batch character set printed depends on the print train used on the line printer to which the file is sent (refer to the ROUTE control statement in section 7). The following are the print trains corresponding to each of the batch character sets.

Character Set	Print Train
CDC graphic 64-character set	596-1
ASCII graphic 64-character set	596-5
ASCII graphic 95-character set	596-6

The characters of the default 596-1 print train are listed in the table 1-A-2 column labeled CDC Graphic (64 Char); the 596-5 print train characters are listed in the table 1-A-2 column labeled ASCII Graphic (64 Char); and the 596-6 print train characters are listed in the table 1-A-2 column labeled ASCII Graphic (95 Char).

If a transmission error occurs when printing a line, the system prints the line again. The CDC graphic print train prints a concatenation symbol (⌐) in the first printable column of a line containing errors. The ASCII print trains print an underline (_) instead of the concatenation symbol.

If an unprintable character exists in a line (that is, a 12-bit ASCII code outside the range 0040_8 through 0176_8), the number sign (#) appears in the first printable column of a print line, and a space replaces the unprintable character.

TABLE 1-A-1. TIME-SHARING CHARACTER SETS

ASCII Graphic (64 Char)	ASCII Character (128 Char)	Display Code	6/12 Display Code	12-Bit ASCII Code	ASCII Graphic (64 Char)	ASCII Character (128 Char)	Display Code	6/12 Display Code	12-Bit ASCII Code
: colon†		00†			# num. sign	# num. sign	60	60	0043
Display code 00 is undefined at sites using the 63-character set.					[l. bracket	[l. bracket	61	61	0133
					] r. bracket	] r. bracket	62	62	0135
A	A	01	01	0101	% †	% †	63†	63†	0045
B	B	02	02	0102	: colon	: colon	63	63	0072
C	C	03	03	0103	" quote	" quote	64	64	0042
D	D	04	04	0104	_ underline	_ underline	65	65	0137
E	E	05	05	0105	!	!	66	66	0041
F	F	06	06	0106	& ampersand	& ampersand	67	67	0046
G	G	07	07	0107					
					' apostrophe	' apostrophe	70	70	0047
					?	?	71	71	0077
H	H	10	10	0110	<	<	72	72	0074
I	I	11	11	0111	>	>	73	73	0076
J	J	12	12	0112	@		74		
K	K	13	13	0113	\ rev. slant	\ rev. slant	75	75	0134
L	L	14	14	0114	^ circumflex		76		
M	M	15	15	0115	; semicolon	; semicolon	77	77	0073
N	N	16	16	0116					
O	O	17	17	0117		@		7401	0100
						^ circumflex		7402	0136
						: colon†		7404†	0072
P	P	20	20	0120		%		7405	0045
Q	Q	21	21	0121		` grave accent		7407	0140
R	R	22	22	0122					
S	S	23	23	0123		a		7601	0141
T	T	24	24	0124		b		7602	0142
U	U	25	25	0125		c		7603	0143
V	V	26	26	0126		d		7604	0144
W	W	27	27	0127		e		7605	0145
						f		7606	0146
X	X	30	30	0130		g		7607	0147
Y	Y	31	31	0131					
Z	Z	32	32	0132		h		7610	0150
0	0	33	33	0060		i		7611	0151
1	1	34	34	0061		j		7612	0152
2	2	35	35	0062		k		7613	0153
3	3	36	36	0063		l		7614	0154
4	4	37	37	0064		m		7615	0155
						n		7616	0156
						o		7617	0157
5	5	40	40	0065					
6	6	41	41	0066		p		7620	0160
7	7	42	42	0067		q		7621	0161
8	8	43	43	0070		r		7622	0162
9	9	44	44	0071		s		7623	0163
+	+	45	45	0053		t		7624	0164
-	-	46	46	0055		u		7625	0165
*	*	47	47	0052		v		7626	0166
						w		7627	0167
/	/	50	50	0057		x		7630	0170
(	(	51	51	0050		y		7631	0171
)	)	52	52	0051		z		7632	0172
$	$	53	53	0044		{ left brace		7633	0173
=	=	54	54	0075		| vert. line		7634	0174
space	space	55	55	0040		} right brace		7635	0175
, comma	, comma	56	56	0054		~ tilde		7636	0176
. period	. period	57	57	0056		DEL		7637	0177

† The interpretation of this character or code depends on its context. Refer to Character Set Anomalies in this appendix.

79AAIA
1 OF 2

TABLE 1-A-1. TIME-SHARING CHARACTER SETS

ASCII Graphic (64 Char)	ASCII Character (128 Char)	Display Code	6/12 Display Code	12-Bit ASCII Code	ASCII Graphic (64 Char)	ASCII Character (128 Char)	Display Code	6/12 Display Code	12-Bit ASCII Code
	NUL		7640	4000		DLE		7660	0020
	SOH		7641	0001		DC1		7661	0021
	STX		7642	0002		DC2		7662	0022
	ETX		7643	0003		DC3		7663	0023
	EOT		7644	0004		DC4		7664	0024
	ENQ		7645	0005		NAK		7665	0025
	ACK		7646	0006		SYN		7666	0026
	BEL		7647	0007		ETB		7667	0027
	BS		7650	0010		CAN		7670	0030
	HT		7651	0011		EM		7671	0031
	LF		7652	0012		SUB		7672	0032
	VT		7653	0013		ESC		7673	0033
	FF		7654	0014		FS		7674	0034
	CR		7655	0015		GS		7675	0035
	SO		7656	0016		RS		7676	0036
	SI		7657	0017		US		7677	0037

79AA1A
2 OF 2

TABLE 1-A-2. BATCH CHARACTER SETS

CDC Graphic (64 Char)	ASCII Graphic (64 Char)	ASCII Graphic (95 Char)	Display Code	6/12 Display Code	12-Bit ASCII Code	Punch Code	
						026	029
: colon†	: colon†		00†			8-2	8-2
Display code 00 is undefined at sites using the 63-character set.							
A	A	A	01	01	0101	12-1	12-1
B	B	B	02	02	0102	12-2	12-2
C	C	C	03	03	0103	12-3	12-3
D	D	D	04	04	0104	12-4	12-4
E	E	E	05	05	0105	12-5	12-5
F	F	F	06	06	0106	12-6	12-6
G	G	G	07	07	0107	12-7	12-7
H	H	H	10	10	0110	12-8	12-8
I	I	I	11	11	0111	12-9	12-9
J	J	J	12	12	0112	11-1	11-1
K	K	K	13	13	0113	11-2	11-2
L	L	L	14	14	0114	11-3	11-3
M	M	M	15	15	0115	11-4	11-4
N	N	N	16	16	0116	11-5	11-5
O	O	O	17	17	0117	11-6	11-6
P	P	P	20	20	0120	11-7	11-7
Q	Q	Q	21	21	0121	11-8	11-8
R	R	R	22	22	0122	11-9	11-9
S	S	S	23	23	0123	0-2	0-2
T	T	T	24	24	0124	0-3	0-3
U	U	U	25	25	0125	0-4	0-4
V	V	V	26	26	0126	0-5	0-5
W	W	W	27	27	0127	0-6	0-6
X	X	X	30	30	0130	0-7	0-7
Y	Y	Y	31	31	0131	0-8	0-8
Z	Z	Z	32	32	0132	0-9	0-9
0	0	0	33	33	0060	0	0
1	1	1	34	34	0061	1	1
2	2	2	35	35	0062	2	2
3	3	3	36	36	0063	3	3
4	4	4	37	37	0064	4	4
5	5	5	40	40	0065	5	5
6	6	6	41	41	0066	6	6
7	7	7	42	42	0067	7	7
8	8	8	43	43	0070	8	8
9	9	9	44	44	0071	9	9
+	+	+	45	45	0053	12	12-8-6
-	-	-	46	46	0055	11	11
*	*	*	47	47	0052	11-8-4	11-8-4

† The interpretation of this character or code depends on its context. Refer to Character Set Anomalies in this appendix.

79AA2A
1 OF 3

TABLE 1-A-2. · BATCH CHARACTER SETS

CDC Graphic (64 Char)	ASCII Graphic (64 Char)	ASCII Graphic (95 Char)	Display Code	6/12 Display Code	12-Bit ASCII Code	Punch Code	
						026	029
/	/	/	50	50	0057	0-1	0-1
(	(	(	51	51	0050	0-8-4	12-8-5
)	)	)	52	52	0051	12-8-4	11-8-5
$	$	$	53	53	0044	11-8-3	11-8-3
=	=	=	54	54	0075	8-3	8-6
space	space	space	55	55	0040	no punch	no punch
, comma	, comma	, comma	56	56	0054	0-8-3	0-8-3
. period	. period	. period	57	57	0056	12-8-3	12-8-3
≡ equiv.	# num. sign	# num. sign	60	60	0043	0-8-6	8-3
[l. bracket	[l. bracket	[l. bracket	61	61	0133	8-7	12-8-2
] r. bracket	] r. bracket	] r. bracket	62	62	0135	0-8-2	11-8-2
% †	% †	% †	63†	63†	0045	8-6	0-8-4
: colon	: colon	: colon	63	63	0072	8-2	8-2
≠	" quote	" quote	64	64	0042	8-4	8-7
↱	_ underline	_ underline	65	65	0137	0-8-5	0-8-5
∨	¬	¬	66	66	0041	11-0	12-8-7
∧	& ampersand	& ampersand	67	67	0046	0-8-7	12
↑	' apostrophe	' apostrophe	70	70	0047	11-8-5	8-5
↓	?	?	71	71	0077	11-8-6	0-8-7
<	<	<	72	72	0074	12-0	12-8-4
>	>	>	73	73	0076	11-8-7	0-8-6
≤	@		74			8-5	8-4
≥	\ rev. slant	\ rev. slant	75	75	0134	12-8-5	0-8-2
¬	^ circumflex		76			12-8-6	11-8-7
; semicolon	; semicolon	; semicolon	77	77	0073	12-8-7	11-8-6
		@		7401	0100		
		^ circumflex		7402	0136		
		: colon †		7404†	0072		
		%		7404	0045		
		` grave accent		7407	0140		
		a		7601	0141		
		b		7602	0142		
		c		7603	0143		
		d		7604	0144		
		e		7605	0145		
		f		7606	0146		
		g		7607	0147		

† The interpretation of this character or code depends on its context. Refer to Character Set Anomalies in this appendix.

79AA2A
2 OF 3

TABLE 1-A-2.· BATCH CHARACTER SETS

CDC Graphic (64 Char)	ASCII Graphic (64 Char)	ASCII Graphic (95 Char)	Display Code	6/12 Display Code	12-Bit ASCII Code	Punch Code	
						026	029
		h		7610	0150		
		i		7611	0151		
		j		7612	0152		
		k		7613	0153		
		l		7614	0154		
		m		7615	0155		
		n		7616	0156		
		o		7617	0157		
		p		7620	0160		
		q		7621	0161		
		r		7622	0162		
		s		7623	0163		
		t		7624	0164		
		u		7625	0165		
		v		7626	0166		
		w		7627	0167		
		x		7630	0170		
		y		7631	0171		
		z		7632	0172		
		{ left brace		7633	0173		
		| vert. line		7634	0174		
		} right brace		7635	0175		
		~ tilde		7636	0176		

79AA2A
3 OF 3

TABLE 1-A-3. ASCII TO 6/12 DISPLAY CODE CONVERSION

ASCII Character (128 Char)	12-Bit ASCII Code		6/12 Display Code	ASCII Character (128 Char)	12-Bit ASCII Code		6/12 Display Code
	Octal	Hex			Octal	Hex	
NUL	4000	00	7640	0	0060	30	33
SOH	0001	01	7641	1	0061	31	34
STX	0002	02	7642	2	0062	32	35
ETX	0003	03	7643	3	0063	33	36
EOT	0004	04	7644	4	0064	34	37
ENQ	0005	05	7645	5	0065	35	40
ACK	0006	06	7646	6	0066	36	41
BEL	0007	07	7647	7	0067	37	42
BS	0010	08	7650	8	0070	38	43
HT	0011	09	7651	9	0071	39	44
LF	0012	0A	7652	: colon †	0072	3A	7404 †
VT	0013	0B	7653	: colon	0072	3A	63
FF	0014	0C	7654	; semicolon	0073	3B	77
CR	0015	0D	7655	<	0074	3C	72
SO	0016	0E	7656	=	0075	3D	54
SI	0017	0F	7657	>	0076	3E	73
				?	0077	3F	71
DLE	0020	10	7660				
DC1	0021	11	7661	@	0100	40	7401
DC2	0022	12	7662	A	0101	41	01
DC3	0023	13	7663	B	0102	42	02
DC4	0024	14	7664	C	0103	43	03
NAK	0025	15	7665	D	0104	44	04
SYN	0026	16	7666	E	0105	45	05
ETB	0027	17	7667	F	0106	46	06
				G	0107	47	07
CAN	0030	18	7670				
EM	0031	19	7671	H	0110	48	10
SUB	0032	1A	7672	I	0111	49	11
ESC	0033	1B	7673	J	0112	4A	12
FS	0034	1C	7674	K	0113	4B	13
GS	0035	1D	7675	L	0114	4C	14
RS	0036	1E	7676	M	0115	4D	15
US	0037	1F	7677	N	0116	4E	16
				O	0117	4F	17
space	0040	20	55				
!	0041	21	66	P	0120	50	20
" quote	0042	22	64	Q	0121	51	21
# number sign	0043	23	60	R	0122	52	22
$	0044	24	53	S	0123	53	23
% †	0045	25	63 †	T	0124	54	24
%	0045	25	7404	U	0125	55	25
& ampersand	0046	26	67	V	0126	56	26
' apostrophe	0047	27	70	W	0127	57	27
(	0050	28	51	X	0130	58	30
)	0051	29	52	Y	0131	59	31
*	0052	2A	47	Z	0132	5A	32
+	0053	2B	45	[left bracket	0133	5B	61
, comma	0054	2C	56	\ reverse slant	0134	5C	75
-	0055	2D	46	] right bracket	0135	5D	62
. period	0056	2E	57	^ circumflex	0136	5E	7402
/	0057	2F	50	_ underline	0137	5F	65

† The interpretation of this character or code may depend on its context. Refer to Character Set Anomalies elsewhere in this appendix.

79AA3A
1 OF 2

TABLE 1-A-3. ASCII TO 6/12 DISPLAY CODE CONVERSION

ASCII Character (128 Char)	12-Bit ASCII Code		6/12 Display Code	ASCII Character (128 Char)	12-Bit ASCII Code		6/12 Display Code
	Octal	Hex			Octal	Hex	
grave accent	0140	60	7407	p	0160	70	7620
a	0141	61	7601	q	0161	71	7621
b	0142	62	7602	r	0162	72	7622
c	0143	63	7603	s	0163	73	7623
d	0144	64	7604	t	0164	74	7624
e	0145	65	7605	u	0165	75	7625
f	0146	66	7606	v	0166	76	7626
g	0147	67	7607	w	0167	77	7627
h	0150	68	7610	x	0170	78	7630
i	0151	69	7611	y	0171	79	7631
j	0152	6A	7612	z	0172	7A	7632
k	0153	6B	7613	{ left brace	0173	7B	7633
l	0154	6C	7614	| vertical line	0174	7C	7634
m	0155	6D	7615	} right brace	0175	7D	7635
n	0156	6E	7616	~ tilde	0176	7E	7636
o	0157	6F	7617	DEL	0177	7F	7637

79AA3A
2 OF 2

MAGNETIC TAPE USERS

Coded data to be copied from mass storage to magnetic tape is assumed to be represented in display code. NOS converts the data to external BCD code when writing a coded 7-track tape and to ASCII or EBCDIC code (as specified on the tape assignment statement) when writing a coded 9-track tape.

Because only 63 characters can be represented in 7-track even parity, one of the 64 display codes is lost in conversion to and from external BCD code. The following shows the differences in conversion depending on the character set (63 or 64) which the system uses.

63-Character Set

Display Code		External BCD		Display Code
00		16		00
33	Output →	12	Input →	33
63		12		33

64-Character Set

Display Code		External BCD		Display Code
00		12		33
33	Output →	12	Input →	33
63		16		63

If a lowercase ASCII or EBCDIC code is read from a 9-track coded tape, it is converted to its uppercase 6-bit display code equivalent. To read and write lowercase ASCII or EBCDIC characters, the user must assign the tape in binary mode and perform his own conversion of the binary data.

Table 1-A-4 lists the uppercase character codes and their display code equivalents. Table 1-A-5 lists the display code equivalents for lowercase character codes read.

TABLE 1-A-4. CODED TAPE CONVERSION (UPPERCASE CHARACTERS ONLY)

Display Code (Octal)	7-Track Tapes	9-Track Tapes		Display Code (Octal)	7-Track Tapes	9-Track Tapes	
	External BCD	ASCII	EBCDIC		External BCD	ASCII	EBCDIC
00	12†	072	172	40	05	065	365
01	61	101	301	41	06	066	366
02	62	102	302	42	07	067	367
03	63	103	303	43	10	070	370
04	64	104	304	44	11	071	371
05	65	105	305	45	60	053	116
06	66	106	306	46	40	055	140
07	67	107	307	47	54	052	134
10	70	110	310	50	21	057	141
11	71	111	311	51	34	050	115
12	41	112	321	52	74	051	135
13	42	113	322	53	53	044	133
14	43	114	323	54	13	075	176
15	44	115	324	55	20	040	100
16	45	116	325	56	33	054	153
17	46	117	326	57	73	056	113
20	47	120	327	60	36	043	173
21	50	121	330	61	17	133	112
22	51	122	331	62	32	135	132
23	22	123	342	63	16†	045	154
24	23	124	343	64	14	042	177
25	24	125	344	65	35	137	155
26	25	126	345	66	52	041	117
27	26	127	346	67	37	046	120
30	27	130	347	70	55	047	175
31	30	131	350	71	56	077	157
32	31	132	351	72	72	074	114
33	12†	060	360	73	57	076	156
34	01	061	361	74	15	100	174
35	02	062	362	75	75	134	340
36	03	063	363	76	76	136	137
37	04	064	364	77	77	073	136

†As explained previously in this section, conversion of these codes depends on whether the tape is being read or written.

TABLE 1-A-5. LOWERCASE CHARACTER CONVERSION FOR READING 9-TRACK CODED TAPES

Display Code	9-Track Tapes		Display Code	9-Track Tapes	
	ASCII	EBCDIC		ASCII	EBCDIC
00	032	077	40	025	075
01	141	201	41	026	062
02	142	202	42	027	046
03	143	203	43	030	030
04	144	204	44	031	031
05	145	205	45	013	013
06	146	206	46	015	015
07	147	207	47	012	045
10	150	210	50	017	017
11	151	211	51	010	026
12	152	221	52	011	005
13	153	222	53	004	067
14	154	223	54	035	035
15	155	224	55	000	000
16	156	225	56	014	014
17	157	226	57	016	016
20	160	227	60	003	003
21	161	230	61	034	034
22	162	231	62	001	001
23	163	242	63	005	055
24	164	243	64	002	002
25	165	244	65	177	007
26	166	245	66	175	320
27	167	246	67	006	056
30	170	247	70	007	057
31	171	250	71	037	037
32	172	251	72	173	300
33	020	020	73	036	036
34	021	021	74	140	171
35	022	022	75	174	152
36	023	023	76	176	241
37	024	074	77	033	047

MESSAGES B

This appendix contains an alphabetical listing of the messages that may appear in a user's dayfile or output file. Lowercase characters identify variable names or fields. Messages beginning with variable names or characters follow those beginning with A through Z and 0 through 9. These messages are alphabetized according to the first nonvariable word or character. Messages beginning with any special characters (such as hyphens or asterisks) are alphabetized as if the special character were not present. For example, the message

pfn ALREADY PERMANENT, AT nnn.

is listed after the messages beginning with A through Z and 0 through 9 and is alphabetized with the messages whose first nonvariable word or character begins with A.

Dayfile messages usually issued only to COMPASS programs are listed in appendix B in volume 2. If a message received during processing of a time-sharing job is not listed in this appendix or in appendix B in volume 2, refer to the IAF Reference Manual or the Time-Sharing User's Reference Manual.

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ACCOUNT BLOCK LIMIT.	The monitor detected the expiration of the account block SRU limit.	Reset account block SRU limit with SETASL control statement or macro. If the account block limit is set at its maximum, issue another CHARGE statement to begin a new account block.	1AJ
ADDRESS OUT OF RANGE addr.	LOC read an address addr on a correction statement that is greater than or equal to the user's field length.	The correction statement is ignored and LOC continues.	CPMEM
ADDRESS OUT OF RANGE.	An address in a parameter block is outside the user's field length.	Specify parameter block address within field length.	LFM
ARG. ERROR.	LDR parameters were outside the user's field length.	Examine program to determine error.	LDR
ARGUMENT ERROR.	A control statement is syntactically incorrect. Refer to the appropriate control statement or command for further information. When the system processes tape management statements, it issues this message if both ring enforcement options (PO=R and PO=W) or more than one option (PO=I, PO=P, and PO=S) is specified. Also, specification of duplicate parameters (more than one occurrence of a keyword) or multiple equivalent parameters (such as MT/NT, CB/CK, FI/L, R/W, and so forth) is not allowed on tape assignment control statements. An address parameter on DMPECS, DMDECS, LBC, LOC, PBC, or WBR must be numeric.	Recheck parameters.	CONVERT, COPY, COPYBR, COPYBF, COPYEI, COPYX, CPMEM, LO72, RESEX, TCOPY,
ARGUMENT ERRORS.	The ENQUIRE control statement is syntactically incorrect.	Check parameters on control statement and retry.	ENQUIRE
ARITHMETIC INDEFINITE.	The CPU floating-point arithmetic unit attempted to use an indefinite operand.	Analyze the job output and dumps to determine the cause.	1AJ
ARITHMETIC OVERFLOW.	The CPU floating-point arithmetic unit received an operand too large for computation.	Analyze the job output and dumps to determine the cause.	1AJ

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ARITHMETIC UNDERFLOW.	The CPU floating-point arithmetic unit received an operand too small for computation.	Analyze the job output and dumps to determine the cause.	!AJ
BAD DECK NAME.	A deck name has more than seven characters.	Correct error and rerun.	UPMOD
BINARY SEQ. ERROR, RECxxxx CDyyyy.	A binary card was found to be out of sequence and the job was terminated. Error is on card yyyy (octal) of record xxxx (octal).	Examine sequencing of binary deck and correct error.	!CD
BLANK TAPE, lfn AT addr.	A blank tape was read. (Blank tape is defined as more than 25 feet of erased tape.)	Ensure correct tape is specified on control statement.	!MT
BLOCK COUNT ERROR IN TRAILER LABEL, lfn AT addr.	The block count in the EOF! or EOV! label did not match the block count maintained by the tape executive during the read operation.	Inform site analyst.	!MT
BLOCK SEQUENCE ERROR, lfn AT addr.	The block length recorded in the file did not match the length of the block read, or the block number recorded in the file did not match the system block count (this message applies to I format tapes only).	Ensure accuracy of format parameter (F) on control statement or macro.	!MT
BLOCK SIZE TOO LARGE ON filenam.	S, L, or F tape block size exceeds copy specifications.	Reduce the block size.	COPY, TCOPY
BLOCK SIZE TOO SMALL ON filenam.	Block size on S, L, or F tape does not meet copy requirements.	Increase block size so that it is greater than the noise size. On F to F tape copy, increase output file block size so it is greater than the block size of the input file.	COPY, TCOPY
BLOCK TOO LARGE, lfn AT addr.	The tape being read contained a data block greater in size than that allowed by the specified format or by user declaration.	Ensure accuracy of format parameter (F) on control statement or macro.	!MT
BOT/EOT ENCOUNTERED, lfn AT addr.	Indicates an abnormal tape position.	Inform site analyst if persistent.	!MT
BREAKPOINT CONDITION.	The job executed an address for which a breakpoint was requested by the system.	Inform site analyst.	!AJ

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
BUFFER ARGUMENT ERROR, lfn AT addr.	For tape operations, this message indicates one of the following. - FET less than 7 words long for S/L format - MLRS greater than 1000 octal for S format - POSMF issued and no HDR1 label found in FET or extended label buffer Refer to volume 2 of the NOS Reference Manual.	Examine program to determine error.	IMT
BUFFER ARGUMENT ERROR ON lfn AT addr.	A buffer pointer did not conform to the following constraints. - FIRST .LE. IN - FIRST .LE. OUT - OUT .LT. LIMIT .LE. FL Refer to volume 2 of the NOS Reference Manual.	Examine program to determine error in buffer pointers.	CIO
BUFFER CONTROL WORD ERROR.	Dayfile message indicating that the word count in the disk linkage is greater than 100B.	Inform site analyst.	SLL
BUFFER CONTROL WORD ERROR, lfn AT addr.	Either an attempt was made to write a block smaller than the noise size on an S, L, or F format tape, or a control word error occurred in a write (such as bad byte count). Refer to volume 2 of the NOS Reference Manual.	Examine program to determine error.	IMT
BUFFER TOO SMALL.	The buffer in the DAYFILE is not large enough to contain a copy of the system dayfile buffer.	None.	DAYFILE
CATALOG COMPLETE.	Informative message indicating that cataloging or the list run is complete.	None.	CATALOG, MODVAL, PFATC
CATALOG FILE NAME CONFLICT.	The same file was named as the file to be cataloged and as the file onto which the catalog is written.	Change one of the file names on the CATALOG control statement.	CATALOG
CATALOG OVERFLOW - FILES, AT addr.	The number of files in the user's catalog exceeds his limit.	One or more permanent files must be purged in order to save or define additional files.	PFM
CATALOG OVERFLOW - SIZE AT addr.	The cumulative size of the indirect access files in the user's catalog exceeds his limit.	One or more indirect access files must be purged or shortened to allow additional	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
		permanent file space.	
CCL100-SEPARATOR FOLLOWING VERB MUST BE COMMA OR LEFT PARENTHESIS	Fatal user error. Separator following verb in a CCL statement must be a comma or a left parenthesis.	Change separator following verb to a comma or a left parenthesis.	CCL
CCL101-LAST NON-BLANK CHARACTER MUST BE SEPARATOR	Fatal user error. Last character string of card or line was not followed by a separator or a terminator.	To terminate statement, make last nonblank character a period or right parenthesis. To continue statement on next card or line, make last nonblank character a valid separator.	CCL
CCL102-EQUAL SIGN MUST FOLLOW FIRST SYMBOLIC NAME	Fatal user error. First parameter following a SET verb is a symbolic name to be set. An equal sign must follow the symbolic name.	Change separator following the symbolic name to an equal sign. Equal sign must be followed by an expression.	CCL
CCL103-STATEMENT INCOMPLETE	Fatal user error. A terminator was detected immediately following a verb.	Check statement format (refer to section 4 of NOS Reference Manual, volume 1) and rewrite statement, using a comma or left parenthesis after verb.	CCL
CCL104-EXPRESSION AND STATEMENT TERMINATED BY t	Fatal user error. A CCL expression or verb was followed by a statement terminator (period or right parenthesis) instead of a comma. Any of following conditions may produce this error. This message is informative and is issued in conjunction with message CCL123. - The label string parameter was omitted. - A relational or logical operator was delimited by a period on left side only. - An unbalanced right parenthesis appeared in expression. If separator following verb was a left parenthesis, an unbalanced right parenthesis may appear balanced. For example, IFE(R1+R2)=3, LABEL. is terminated by right parenthesis.	- Place a comma after expression or verb and add label string parameter. - Delimit operator with periods on both sides. - If error was caused by a misleading left parenthesis following verb, add a comma immediately after verb. Otherwise recheck all parentheses and balance unbalanced right parenthesis.	CCL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
CCL105-TERMINATOR MUST FOLLOW SUBSYSTEM NAME	Fatal user error. The SS function must be of the form SS. or SS=name.	Place a period in the appropriate location.	CCL
CCL120-ALL CARDS SKIPPED - xxxxxx	Fatal user error. In process of skipping, end of control statement record was reached. Skipping was initiated by a CCL statement in either control statement file (ENDRUN) or a procedure (REVERT). Any of following conditions may produce this error. - An IFE, SKIP, or ELSE expression was false and no ENDIF statement terminated resultant skip. - A WHILE expression was false and no ENDW statement terminated resultant skip. - An ENDW statement was not preceded by a WHILE statement.	- Either add an ENDIF statement with a matching label string or correct existing ENDIF statement. - Either add an ENDW statement with a matching label string or correct existing ENDW statement so that WHILE and ENDW label strings match. - Precede ENDW statement with a WHILE statement.	CCL
CCL121-LABEL STRING MUST BEGIN WITH ALPHA CHARACTER	Fatal user error. First character of a label string must not be numeric.	Change label string to a label string of 1 to 7 alphanumeric characters, beginning with an alphabetic character.	CCL
CCL123-ALPHANUMERIC LABEL STRING REQUIRED	Fatal user error. CCL statements ELSE, ENDIF, ENDW, IFE, SKIP, and WHILE require label string parameter. This message is issued in conjunction with message CCL104.	Refer to message CCL104 for cause and suggested action.	CCL
CCL124-PRECEDING ERR MSG. APPLIES TO FOLLOWING	Informative message. Dayfile message which precedes CCL124 is an error message which was caused by the statement printed after CCL124.	Refer to error message which precedes this in the dayfile for cause and suggested action.	CCL
CCL125-NUMBER OF CARDS SUPPRESSED=x	Informative message. A statement which CCL is printing from an internal buffer continued over more cards or lines of input than buffer size. CCL prints first three cards or lines and then prints this message to indicate number of cards or lines suppressed, x. Remaining cards or lines of statement follow this message.	None.	CCL
CCL126-STATEMENT TERMINATOR MUST FOLLOW LABEL	Fatal user error. Either of the following conditions may produce this error. - A label string is last parameter on an ELSE, ENDIF, ENDW, IFE, SKIP, or WHILE	- Terminate CCL statement with a period or a right parenthesis.	CCL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	statement. It must be followed by a period or a right parenthesis. - A CCL expression contained a comma, and character string that followed was assumed to be a label string.	- Either remove comma or replace it with a left parenthesis, depending on nature of error.	
CCL127-LABEL STRING EXCEEDS xx CHARACTERS	Fatal user error. Label string parameter on an ELSE, ENDIF, ENDW, IFE, SKIP, or WHILE statement is greater than xx characters. xx is maximum number of characters defined by installation.	Change label string on any CCL statement it appears to a label string with xx or fewer alphanumeric characters.	CCL
CCL150-EXPRESSION CONTAINS INVALID OPERATOR x	Fatal user error. x is not a legal operator in preceding CCL statement.	If x is part of a character string, $-delimit character string. If x is an intended operator, refer to section 4 for a list of valid operators.	CCL
CCL151-EXPRESSION FORMAT ERROR (IN FILE FUNCTION)	Fatal user error. Format of expression is illegal. Probable cause is use of commas or unbalanced left parentheses within an expression. If error was detected within a CCL FILE function, IN FILE FUNCTION is included in message. FILE function must be of form FILE(lfn,exp), where exp is a valid FILE function.	Check parentheses and balance unbalanced left parenthesis or remove excess commas from FILE function.	CCL
CCL152-POORLY FORMED FUNCTION (NUM, DT, SS)	Fatal user error with following causes. - Separator after CCL function name is not a left parenthesis or function is not terminated by a right parenthesis. - Character string evaluated by CCL function contained special characters but was not $-delimited.	- Correct format of function. - $-delimit character string.	CCL
CCL153-ILLEGAL FUNCTION CALL WITHIN FILE FUNCTION	Fatal user error. A CCL NUM, SS, or FILE function was called from within a FILE function.	Restructure CCL expression so NUM, SS, or FILE function is no longer within a FILE function.	CCL
CCL154-ILLEGAL EXPONENT	Fatal user error. A negative exponent in a CCL expression is illegal.	Remove minus sign preceding exponent.	CCL
CCL155-OPERATOR OR OPERAND SEQUENCE ERROR	Fatal user error with following causes. - Sequence of adjacent operators is illegal. For example, 3*-4 is not allowed. - An operand is adjacent to a left or right parenthesis. Implied	- Use parentheses to separate adjacent operators. - Use an operator to separate operand and parenthesis.	CCL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	multiplication is not allowed. For example, 3(R+1) is illegal.		
CCL156-STRING TOO LONG -strng	Fatal user error. Listed character string is too long for a CCL expression. A character string may not be longer than 10 characters, excluding $ delimiters for literals.	Shorten character string to 10 characters or less.	CCL
CCL157-UNKNOWN NAME -strng	Fatal user error. Alphanumeric character string strng is not a symbolic name recognized by CCL.	Replace strng with a valid symbolic name. Refer to Section 4 for a list of valid symbolic names.	CCL
CCL158-OPERATOR/TERMINATOR MUST FOLLOW FUNCTION (DT, FILE, NUM)	Fatal user error. CCL function was properly formed and positioned within CCL statement, but it was not followed by an operator, separator, or terminator.	Place an operator, separator, or terminator after function.	CCL
CCL159-FIRST PARAMETER INVALID IN SET STATEMENT	Fatal user error. First parameter in SET statement was not one of following symbolic names: DSC, EF, EFG, R1, R2, R3, R1G.	Replace first parameter with a valid symbolic name.	CCL
CCL160-NUMERIC OR LITERAL IN FILE FUNCTION	Fatal user error. Expression of a FILE function may contain only symbolic names defined for function.	Remove numeric or literal character string. Refer to Section 4 for a list of FILE symbolic names.	CCL
CCL161-STACK OVERFLOW	Fatal user error. Evaluation of an expression overflowed either operator or operand stack.	Check expression for errors and correct. If no errors, either simplify expression or break it apart so that two or more CCL statements have same effect as one.	CCL
CCL163-SUBSYSTEM REFERENCE ERROR	A name referenced by the SS function does not exist in table of names known to CCL.	Refer to section 4 for a list of valid names.	CCL
CCL200-PROCEDURE NESTING LEVEL xx EXCEEDED	Fatal user error. Current procedure call forced procedure nesting to exceed limit of xx, which is defined by installation.	Reposition procedure call statements so limit xx is not exceeded.	CCL
CCL201-PROCEDURE NAME MORE THAN 7 CHARACTERS	Fatal user error. Procedure name cannot be greater than 7 characters.	Rename procedure.	CCL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
CCL203-PROCEDURE FILE NAME NOT SPECIFIED OR INVALID	Fatal user error with following causes. - File name specified on BEGIN statement was greater than 40 characters. - pfile parameter on BEGIN statement was null, indicating default file name. CCL was installed with default file flag turned off; no default file name is allowed.	- Specify a file name with 40 or fewer characters. - Specify a file name.	CCL
CCL204-MULTIPLE EQUIVALENCE SPECIFICATIONS FOR xx	Nonfatal user error. A format keyword has been specified more than once on procedure call statement. Last definition prevails.	If first specification is desired, remove second specification. If second specification is desired, no action is required.	CCL
CCL205-FORMAL PARAMETER LIST DOES NOT INCLUDE -x	Fatal user error. While in equivalence mode, CCL discovered a formal keyword x on procedure call statement that was not specified on header statement.	Remove formal keyword x from procedure call statement.	CCL
CCL206-SYMBOLIC SPECIFICATION INVALID xxx+	Fatal user error. A parameter, xxx, on procedure call statement is followed by a plus sign. Plus sign indicates that CCL is to convert value to display code, and is valid only when preceded by a symbolic name and followed by a D, B, or a null field. (Example: R1+B is a legal value, but 37+R1 is not.)	If plus sign is part of a character string, $-delimit character string. If plus sign should convert a symbolic name to display code, replace xxx with a valid symbolic name. Refer to section 4 of NOS Reference Manual, volume 1 for a list of valid symbolic names.	CCL
CCL207-PROCEDURE NAMED BEGIN IS INVALID	Fatal user error. A procedure must not be named BEGIN.	Select another procedure name.	CCL
CCL211-SPECIFICATION EXCEEDS xx CHARACTERS	Fatal user error. Value on a procedure call statement or default value on a procedure header statement is greater than xx characters. xx is defined by installation.	Specify a value with xx or fewer characters.	CCL
CCL212-SEPARATOR INVALID strng s	Fatal user error. s is illegal separator and strng is character string that precedes s. Any of following conditions produces this error. - On a procedure call statement, the separator preceding a formal keyword or a positionally specified value is not a comma.	- Change separator s to a comma. - $-delimit character string. - Remove #DATA or #FILE from procedure call statement and specify on procedure header	CCL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	- Invalid separator is part of a character string. If a valid separator immediately precedes illegal separator s, the string is null. - #DATA or #FILE has been specified on the procedure call statement.	statement.	
CCL230-PROCEDURE FILE NOT FOUND	Fatal user error. No local file was found with file name indicated on BEGIN statement. Automatic retrieval of a permanent file with that name was inhibited.	Check file name for errors and correct. If name is correct, retrieve file prior to BEGIN statement.	CCL
CCL231-PROCEDURE NOT FOUND	Fatal user error. Local or permanent file indicated on BEGIN statement was found, but CCL could not find procedure on the file.	Check procedure name for errors and correct. If name is correct, check file for procedure.	CCL
CCL234-UNABLE TO LOCATE LIBRARY PARTITION xxx	Fatal user error. CCL could not find procedure xxx, specified on call-by-name statement, on currently defined library set.	Check procedure name for errors and correct. If name is correct, check library for procedure.	CCL
CCL235-FORMAL PARAMETER GT xx CHARACTERS	Fatal user error. Number of characters in a formal keyword on header statement exceeds xx, as defined by installation.	Define a formal keyword with xx or fewer characters.	CCL
CCL236-SPECIAL DEFAULT SPECIFICATION UNKNOWN	Fatal user error. Equivalence symbol, which specifies a special default option on procedure header statement, must be followed by a known keyword of either FILE or DATA.	If equivalence symbol is part of a character string, $-delimit character string. If a special default option is desired, follow equivalence symbol with either DATA or FILE.	CCL
CCL237-SEPARATOR FOLLOWING SECOND DEFAULT IS */*	Fatal user error. In form fk=default1/ default2/ on procedure header statement, second / is illegal.	If / is part of default2, $-delimit character string. If not, either remove it or replace it with a comma or period.	CCL
CCL238-FORMAL PARAMETER LIMIT xx EXCEEDED	Fatal user error. Number of formal parameters on procedure header statement has exceeded xx, as defined by installation.	Remove formal parameters from procedure header statement until xx or fewer parameters remain.	CCL
CCL239-PROCEDURE HEADER NOT TERMINATED	Fatal user error. Procedure header statement was not terminated by a period, and no control statements were found after header statement.	Terminate header statement with a period, and add at least one control statement to	CCL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
		procedure.	
CCL249 - DATA FILE LFN EXCEEDS 7 CHARACTERS - lfn	The file name lfn specified on a .DATA command is longer than seven characters.	Specify a shorter file name.	CCL
CCL250-CONCATENATED STRING EXCEEDS 80 CHARACTERS	Fatal user error. Use of a right arrow (or underline character in ASCII) produced a linked string of more than 80 characters, or a string of 80 characters with a following separator.	Reduce number of linked characters.	CCL
CCL251-DATA COMMAND SPECIFIED CCL FILE-filenam	Fatal user error. File name that user specified on a .DATA command is a CCL working file.	Select another file name.	CCL
CCL252-PROCEDURE CONTAINS NO CONTROL STATEMENTS	Informative message. A procedure should contain at least one control statement.	None.	CCL
CCL261-ENDW WITH MATCHING LABEL FOUND	Nonfatal user error. CCL encountered an ENDW statement in the job control statement record before finding a WHILE statement with a matching label string. A search is initiated for a corresponding WHILE statement. If a WHILE statement with a matching label string is found and WHILE expression is true, normal processing continues with statement following WHILE statement. If WHILE expression is false or if no WHILE statement with a matching label string is found, CCL skips all remaining statements in control statement record.	Check for a WHILE statement and if not present, add. If present, see if WHILE and ENDW label string match and are correct. If they match, place WHILE statement before ENDW statement.	CCL
CCL262-CONTINUING SEARCH, PRINTING SKIPPED CARDS	Informative message. To assist in isolation of WHILE statement errors such as CCL261, all skipped statements will be printed in user's dayfile until WHILE search is complete.	None.	CCL
CCL270-ERR IN CCL WORK FILES, REVERT TO JOB FILE	Fatal user error. An error was encountered in a file used by CCL because of user manipulation of procedures or data. An attempt is made to return to user's assigned job file. If unsuccessful, the job terminates. If successful, job aborts and system searches for an EXIT statement. CCL issues this message if a user includes a CLEAR statement or a NEW or OLD statement without the /ND parameter in a nested procedure. The CLEAR, NEW, and OLD statements release all working files.	CCL work files are not designed for user manipulation; user is cautioned against such usage. Rewrite the job and/or procedures without manipulating CCL work files and resubmit. Remove the CLEAR statement from the nested procedure or add a /ND parameter to any NEW or OLD	CCL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
		statement included.	
CCL271-REVERT NOT ALLOWED WITHIN JOB FILE	Fatal user error. A REVERT statement cannot appear in control statements of job file.	Remove REVERT statement.	CCL
CCL272-INVALID REVERT PARAMETER - xxx	Fatal user error. Parameter xxx was used on REVERT statement. REVERT statement allows only ABORT parameter or no parameters.	Remove parameter xxx or replace with ABORT.	CCL
CCL300-NON-NUMERIC CHARACTER WITHIN NUMERIC TERM	Fatal user error. An element of an expression which begins with a numeric character cannot contain a nonnumeric character except B or D as a post radix.	Remove any nonnumeric characters other than a post radix B or D.	CCL
CCL302-8/9 DIGIT CONFLICT WITH POST RADIX OF 8	Fatal user error. An 8 or 9 is illegal in an octal number.	Remove any 8's or 9's if number is octal or remove post radix B if number is decimal.	CCL
CCL303-LITERAL NOT TERMINATED	Fatal user error. A literal which begins with a dollar sign was not terminated by a second dollar sign before end of card or line of input was detected.	Place a second dollar sign at end of character string denoted as a literal.	CCL
CCL304-STRCCL - SCATTER BUFFER HEADER INVALID	Fatal system error. This is an internal CCL problem.	Inform site analyst.	CCL
CHANNEL MALFUNCTION, lfn AT addr.	Hardware malfunction.	Inform site analyst.	IMT
CHARGE ABORTED.	Dayfile message indicating that a central site operator action caused the CHARGE operation to abnormally terminate.	Reenter CHARGE statement.	CHARGE
CHARGE FILE BUSY.	Dayfile message indicating that the file which the system uses to validate charge numbers and project numbers is busy.	Reenter CHARGE statement.	CHARGE
CHARGE ILLEGAL AT THIS HOUR.	Dayfile message indicating that the specified project number cannot be used at this time of day.	Retry during the time the project number is valid.	CHARGE
CHARGE NUMBER EXPIRED.	Dayfile and output file message indicating the charge number expiration date has occurred.	None.	CHARGE
CHECK DAYFILE FOR ERRORS.	Informative message indicating that the user should check the dayfile for errors.	Examine error messages in dayfile.	COPY, PFATC, PFCAT, PFCOPY, PFDUMP,

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
			PFLOAD, TCOPY
CHECKPOINT nnnn COMPLETE.	Indicates that checkpoint nnnn has completed. Issued if only one checkpoint file is present. For a checkpoint operation, more than two checkpoint files or an illegal combination of checkpoint files was specified.	None.	CHKPT
CHECKPOINT nnnn COMPLETED TO filenam.	Indicates that checkpoint nnnn has been completed to file filenam. Issued if alternate CB checkpoint files are used.	None.	CHKPT
CHECKPOINT FILE ERROR.	During a restart operation, either the checkpoint file specified on the RESTART control statement was empty or RESTART detected a format error during an attempt to read the specified checkpoint file.	Examine checkpoint file to determine error.	CHKPT, RESTART
CHECKPOINT NOT FOUND.	The specified checkpoint (nn parameter on RESTART statement) could not be found on the file.	Verify that checkpoint is on file.	RESTART
CIO ERROR.	Updating of resource file returned error status other than end-of-device.	Inform site analyst.	RESEX
CKP REQUEST.	A checkpoint has been initiated.	None.	CHKPT
CM BLOCK OUT OF RANGE.	Data transfer from ECS specified a CM address outside the job field length.	Analyze the job output and dumps to determine the cause.	IAJ
CM NOT VALIDATED.	The number of CM words specified on the job statement exceeds that for which the user is validated.	User should rerun the job with a CM specification within his limits.	ACCFAM
CM OR EC REQUEST EXCEEDS MAXIMUM.	The user requested a CM field length greater than 377777B or an ECS field length greater than 7777B blocks.	Rerun the job with smaller field length request.	ACCFAM
CM OUT OF RANGE.	The program referenced an address outside the job CM field length.	Analyze job output and dumps to determine the cause.	IAJ
CM PARITY ERROR.	Double data parity error (two data bits failed) between central memory control (CMC) and CM as detected by the single-error correction double-error detection (SECDED) network, or a single parity error when operating in default mode (SECDED network	Inform customer engineer.	IAJ

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	disabled).		
CMC PARITY ERROR.	The CPU sent the central memory control (CMC) data or an address having incorrect parity.	Inform customer engineer.	1AJ
CONFLICTING PARAMETERS.	Input queue type entered more than once.	Reenter with correct parameters.	SUBMIT
CONFLICTING RESOURCE TYPES.	PE, HD, and GE resources cannot be specified concurrently with an NT resource in the same job.	Correct the RESOURC control statement.	RESEX
CONNECT REJECT, lfn AT addr.	Unable to connect unit.	Inform site analyst.	1MT
CONTROL CARD ERROR.	An illegal or invalid parameter was specified on the control statement.	Correct control statement and retry.	EDIT, LISTLB
CONTROL STATEMENT LIMIT.	The user has entered too many control statements.	Reduce the number of control statements to a value within the specified limit. Refer to LIMITS command.	TCS
CONTROLLED BACKSPACE ERROR, lfn AT addr.	Controlled backspace operation failed during write error recovery. Position of tape is uncertain.	Inform customer engineer.	1MT
CONVERSION NOT SUPPORTED.	The PC and NC parameters on the FCOPY control statement specify an unsupported conversion.	Refer to the description of the FCOPY statement for valid PC and NC values.	FCOPY
COPY COMPLETE.	Copy termination condition was satisfied before EOI was encountered.	None.	COPY, COPYBR, COPYBF, COPYCF, COPYCR, COPYSBF, COPYX, TCOPY
COPY FL ABOVE USER LIMIT.	Field length for L or F tape copy exceeds user's current maximum.	Use smaller block size on L and F tapes or increase maximum FL.	COPY
COPY INDETERMINATE.	Copy with S, L, or F tape is unpredictable. Only the COPY utility supports these formats.	Retry copy using the COPY control statement.	COPYBF, COPYBR, COPYEI, COPYX

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
CORE OVERFLOW, JOB ABORTED.	The job field length is too small to hold the tables required to process the program library specified on the UPMOD control statement.	Increase field length and rerun.	UPMOD
CPxx,....	Refer to description of corresponding message beginning with EQ.		
CPM - ARGUMENT ERROR.	Error(s) encountered and job aborted.	Determine error and rerun job step.	CPM
CPM - ILLEGAL PACKNAM.	An illegal pack name has been specified.	Ensure that legal pack name is used.	CPM
CPM - ILLEGAL REQUEST.	A CPM function was issued without the auto recall specified or job was not of system origin. This message is also issued following a request for SRU limit between 163840 and 262082.	Specify auto recall on RA+1 call to CPM, make system origin, or correct SETASL or SETJSL statement.	CPM
CPM - ILLEGAL USER ACCESS.	The user tried to perform a CPM operation for which he is not validated.	None.	CPM
CPU ERROR EXIT AT addr.	The errors listed after this message occurred at address addr, causing job termination.	Refer to the descriptions of the error messages issued with this message.	1AJ
CRxx,....	Refer to description of corresponding message beginning with EQ.		
CUMULATIVE LIMIT EXCEEDED.	Dayfile and output file message indicating that one of the installation-defined resource usage accumulators for this project exceeded the maximum allowed. The system does not update these accumulators in PROFILa. Each installation must provide this capability if desired.	None.	CHARGE
CUMULATIVE SRU LIMIT EXCEEDED.	Dayfile and output file message indicating that accumulated SRUs have exceeded the maximum allowed.	None.	CHARGE
DATA BASE ERROR.	Dayfile message indicating that the system has detected an error in its validation file.	Contact installation personnel.	CHARGE, MODVAL
DATA BASE ERROR n - NOTIFY ANALYST.	System error dayfile message indicating that an abnormal situation exists. n is displayed for consideration by the analyst. The internal documentation, obtained by	Inform site analyst.	PROFILE

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	using the DOCMENT control statement, contains an explanation of each error n for use by the analyst. (Refer to section 7 in volume 1 of the NOS Reference Manual for a description of DOCMENT.)		
DATA/PERMIT ERRORS, lfn AT addr.	Errors were encountered in both data and permit information for file lfn.	Inform site analyst.	PFM
DATA TRANSFER ERROR, AT addr.	An error occurred in a read operation during a file transfer.	Inform site analyst.	PFM
DExx,Ccc,1,sec,ann,Stttt,Aaddr.	An error has been detected on extended core storage. Refer to appendix B of the NOS Operator's Guide for further information.	Inform site analyst.	6DE
DEMAND EXCEEDED.	The user attempted to assign more units than were scheduled on the RESOURC statement.	Increase appropriate parameter value on RESOURC statement.	RESEX
DEMAND FILE ERROR.	Resource execution error was encountered. This error occurred because the demand file (RSXDid) entry does not match the job name.	Inform site analyst.	RESEX
DEMAND VALIDATION ERROR.	The specified number of units exceeds the user's validation limits.	Decrease appropriate parameter value on RESOURC statement.	RESEX
DEVICE ERROR ON FILE lfn AT addr.	An irrecoverable error occurred on the mass storage device containing the file lfn.	Inform site analyst.	CIO
DEVICE UNAVAILABLE, AT addr.	Access to the permanent file device requested is not possible. User may have attempted to access files on a device not present in the alternate system.	Determine that device can be made available by system operator and retry.	PFM
DI....	Refer to the description of corresponding message beginning with EQ.		
DIxx,Ccc,1,sec,ann,Stttt,FNqqqq. or DIxx,Ccc,1,sec,ann,Stttt,Uuu Cyyyy Sttss.	An error has been detected on mass storage device with EST ordinal xx. Refer to appendix B of the NOS Operator's Guide for further information.	Inform site analyst.	7DI
DIRECT ACCESS DEVICE ERROR, AT addr.	The file specified already exists on a device other than the device requested or an illegal device type was specified. The device on which the file resides may not contain direct access files because of one of the following reasons. - The device is not specified as a direct access device in the catalog descriptor	Specify correct device type.	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	table. - The device is not specified as ON and initialized in the catalog descriptor table. - The device is a dedicated indirect access permanent file device. If on an alternate system, the user's master device may not have been transferred to that system.		
DIRECTIVE CARD ERROR.	A LIBEDIT directive has incorrect syntax.	Examine the LIBEDIT output to determine cause of error.	LIBEDIT
DIRECTIVE ERRORS.	Dayfile message indicating that one or more input directives were in error. Fatal error.	Examine output file to determine reason for error.	MODIFY, OPLEDIT, LIBTASK, MODVAL, PROFILE, SYSEDIT
DISPLAY DUMP NOT ALLOWED TO TERMINAL.	A time-sharing user has attempted to enter DMD or DMDECS control statements or DMD or DED system requests without assigning file OUTPUT to a mass storage device.	Assign file OUTPUT to mass storage via ASSIGN control statement or macro and retry.	CPMEM
DJ...	Refer to description of the corresponding message beginning with DI or EQ.		
DK...	Refer to description of the corresponding message beginning with DI or EQ.		
DL...	Refer to description of corresponding message beginning with DI or EQ.		
DM...	Refer to description of corresponding message beginning with DI or EQ.		
DPxx,Ccc,1,sec,ann,Stttt,FNqqqq. or DPxx,Ccc,1,sec,ann,Stttt,Aaddr,Wwwww DPxx,Ccc,1,Gggg...g. DPxx,Ccc,1,Bbbb...b. or DPxx,Ccc,1,sec,ann,Stttt,Aaddr,Wwwww DPxx,Ccc,1,ddd...d.	An error has been detected on distributive data path (DDP). Refer to appendix B of the NOS Operator's Guide for further information.	Inform site analyst.	6DP
DQ...	Refer to description of corresponding message beginning with DI or EQ.		

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
DSP - CAN NOT ROUTE JOB INPUT FILE.	The job input file cannot be routed.	Copy job input file to a local file to be routed.	DSP
DSP - COMPLETE BIT ALREADY SET.	The complete bit was not cleared before DSP was called.	Clear complete bit before calling DSP.	DSP
DSP - DEVICE UNAVAILABLE.	DSP attempted to create a file on a device that was turned off or is currently unavailable for access.	Specify different device or contact site operator.	DSP
DSP - FILE NAME ERROR.	An attempt was made to create a file with an invalid file name.	Specify valid file name.	DSP
DSP - FILE NOT ON MASS STORAGE.	An attempt was made to route a file not on mass storage.	Copy file to mass storage before routing.	DSP
DSP - FILE ON REMOVABLE DEVICE.	A file on a removable device cannot be routed.	Copy file to non-removable device before routing.	DSP
DSP - FNT/DEVICE FULL.	There is no space in the FNT or on the device for current use.	Retry route at a later time.	DSP
DSP - FORMS CODE NOT ALPHANUMERIC.	Forms code must consist of two alphanumeric characters.	Specify alphanumeric forms code.	DSP
DSP - I/O SEQUENCE ERROR.	A request was made on a busy file.	Wait until file is not busy.	DSP
DSP - ILLEGAL FILE MODE.	An attempt was made to route an execute only file.	Verify that the file to be routed is not an execute only file.	DSP
DSP - ILLEGAL FILE TYPE.	The file being processed is not a print, punch, input, or output file type.	Ensure that file being processed is of correct type.	DSP
DSP - ILLEGAL ORIGIN TYPE.	DSP cannot route the file to the input queue with the origin type specified by the caller.	Specify valid origin type.	DSP
DSP - ILLEGAL REQUEST.	One of the following. - DSP was not called with recall (does not apply when queue priority is greater than MXPS). - Parameter list address was out of range. - RA+1 call was formatted incorrectly.	Specify auto recall with DSP call or determine why parameter list address is out of range.	DSP

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
DSP - ILLEGAL USER CARD.	User attempted to route a file with an illegal USER statement to the input queue.	Ensure that valid user number is being used.	DSP
DSP - IMMEDIATE ROUTING - NO FILE.	The specified file for the immediate routing could not be found.	Ensure that file to be routed is available to job for processing.	DSP
DSP - INVALID DISPOSITION CODE.	Specified disposition code is not recognized.	Verify disposition code.	DSP
DSP - INVALID EXTERNAL CHARACTERISTICS.	Caller specified an undefined external characteristic code.	Verify external characteristic code.	DSP
DSP - INVALID TID.	One of the following. - User number and family name parameters were not in CM field length. - TID is greater than or equal to IDLM for batch jobs. - User number specified in parameter block does not compare with user number in control point area.	Verify that TID parameters are valid.	DSP
DSP - LOCAL FILE LIMIT.	User has exceeded his local file validation limits.	Return one or more local files to the system.	DSP
DSP - OUTPUT FILE LIMIT.	Caller has exceeded his output file validation.	If possible, split job into two or more jobs and retry. Otherwise, reduce number of files by copying output to single file and then routing the file.	DSP
DSP - ROUTE TO INPUT NOT IMMEDIATE.	Routing a file to the input queue must be immediate.	Change to immediate route.	DSP
DSP - THIS ROUTING NOT ALLOWED.	An attempt was made to change the origin type or queue type of a deferred routed file.	Rescind prior routing using DC=SC parameter before changing origin or queue type.	DSP
DSP - TOO MANY DEFERRED BATCH JOBS.	User has more jobs in the system than allowed. This check is ignored for users with system origin privileges.	Try at a later time.	DSP
****DUPLICATE CHARGE NUMBER.	Output file message indicating that an existing charge number was referenced on a create run.	Rerun using correct charge number, if required.	PROFILE

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
DUPLICATE COMMON FILE NAME.	A file of the same name as that specified in a COMMON request already exists.	Use different name in request.	LFM
DUPLICATE FILE NAME.	The file specified already exists in the system.	Use different name in request.	LFM
DUPLICATE LINES.	Lines being dumped during a DMP operation were duplicated and suppressed.	None.	CPMEM
****DUPLICATE PROJECT NUMBER.	Output file message indicating that an existing project number was referenced on a create run.	Rerun using correct project number, if required.	PROFILE
DUPLICATE USER NUMBER.	Output file message indicating that the user number encountered on a create run is a duplicate of a user number previously entered. The first entry is used.	Rerun the corrected job or correct the new validation file, if necessary.	MODVAL
EC NOT VALIDATED.	The number of ECS blocks specified on the job statement exceeds that for which the user is validated.	User should check his validation with the LIMITS statement.	ACCFAM
ECS BLOCK OUT OF RANGE.	Data transfer between CM and ECS specified an ECS address outside the job field length.	Analyze the job output and dumps to determine the cause of the error.	1AJ
ECS FLAG REGISTER PARITY.	Parity error detected on ECS flag register operation.	Inform customer engineer.	1AJ
ECS OUT OF RANGE.	Job referenced ECS address outside job field length.	Analyze the job output and dumps to determine the cause of the error.	1AJ
ECS READ ERROR.	An unrecoverable ECS read error occurred.	None.	CPMEM
EMPTY CATALOG.	No entries are present in the catalog.	None.	CATLIST, NDA
EMPTY MSORT INPUT FILE.	Dayfile message indicating that the file specified on the SORT control statement contains no data.	Correct and rerun.	MSORT
END OF INFORMATION ENCOUNTERED.	An end-of-information mark was encountered on the input file.	None.	COPYCF, COPYCR, COPYSBF
END OF TAPE, lfn AT addr.	The end of tape was encountered.	Ensure that correct file manipulation operation is specified.	1MT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ENQUIRY COMPLETE.	Message issued when processing of ENQUIRE control statement is completed.	None.	ENQUIRE
ENTRY POINT NOT FOUND.	The specified entry point could not be found.	Verify that entry point is valid.	1AJ
EOF ENCOUNTERED.	End-of-file was encountered before copy termination condition was satisfied.	None.	COPYX
EOF ENCOUNTERED BEFORE TERMINATION.	An end-of-file was encountered on a CONVERT input file before the specified record count was reached.	Ensure accuracy of input file.	CONVERT
EOI CHANGED BY RECOVERY, lfn AT addr.	The length of file lfn was altered by the mass storage recovery process because TRT EOI did not match EOI on disk.	ATTACH the file with NA option or use CHANGE with CE option to clear error status.	PFM
EOI ENCOUNTERED.	End-of-information was encountered on the input file.	None.	COPY, COPYBR, COPYBF, COPYEI, COPYX, TCOPY
EOI ENCOUNTERED BEFORE TERMINATION.	An end-of-information was encountered on a CONVERT input file before the specified record count was reached.	Ensure accuracy of input file.	CONVERT
EQxx, CHcc Fffff FUNCTION TIMEOUT.	No response (inactive) was received after a function code was issued to the specified local batch equipment (converter and equipment status unavailable). EQ One of the following equipment types. CP 415 card punch CR 405 card reader LP Any line printer LR 580-12 line printer LS 580-16 line printer LT 580-20 line printer xx EST ordinal of local batch equipment cc Channel number ffff Function code	Inform customer engineer.	QAP, 1CD
EQxx, CHcc Fffff REJ Paaaa,Cbbbb,Emmmm.	Function reject or transmission parity error was detected on the specified local batch equipment. EQ One of the following equipment types. CP 415 card punch CR 405 card reader LP Any line printer	Inform customer engineer.	QAP, 1CD

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	LR 580-12 line printer LS 580-16 line printer LT 580-20 line printer xx EST ordinal of local batch equipment cc Channel number ffff Function code aaaa Driver (1CD) address bbbb Converter status mmmm Equipment status		
EQxx, CHcc, PRINT ERROR LIMIT EXCEEDED.	Maximum number of consecutive print errors was detected on line printer xx. EQ One of the following equipment types. LP Any line printer LR 580-12 line printer LS 580-16 line printer LT 580-20 line printer xx EST ordinal of line printer cc Channel number	Inform customer engineer.	1CD, QAP
EQxx, CHcc RESERVED.	The specified local batch equipment is reserved and cannot be connected on channel cc. EQ One of the following equipment types. CP 415 card punch CR 405 card reader LP Any line printer LR 580-12 line printer LS 580-16 line printer LT 580-20 line printer xx EST ordinal of local batch equipment cc Channel number	Inform customer engineer.	1IO
EQxx,DNdn, DIRECT ACCESS FILE ERROR, AT addr.	The system sector data for the file does not match the catalog data. Error log and dayfile message. EQ One of the following equipment types. DI 844-21 disk (half track) DJ 844-41/44 disk (half track) DK 844-21 disk (full track) DL 844-41/44 disk (full track) DM 885 disk (half track) DQ 885 disk (full track) xx EST ordinal of device dn Device number	Inform site analyst.	PFM
EQxx,DNdn, FILE LENGTH ERROR, AT addr.	The length of a file does not equal the length specified in the catalog. EQ Equipment type as defined in EQxx,DNdn, DIRECT ACCESS FILE ERROR, AT addr. message	Inform site analyst.	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	xx EST ordinal of device dn Device number The cause depends on the type of command or macro issued. GET — A local file is created with length being the actual length retrieved. SAVE — If file length is longer than TRT specification, file is truncated. REPLACE — Same as for SAVE.		
EQxx,DNdn, MASS STORAGE ERROR AT addr.	An error was encountered in reading a portion of the permanent file catalog or permit information. Error log and dayfile message. EQ Equipment type as defined in the EQxx,DNdn, DIRECT ACCESS FILE ERROR, AT addr. message xx EST ordinal of device dn Device number	Inform site analyst.	PFM
EQxx,DNdn, RANDOM INDEX ERROR, AT addr.	The random disk address of the permit sector is in error. Error log and dayfile message. EQ Equipment type as defined in the EQxx,DNdn, DIRECT ACCESS FILE ERROR, AT addr. message xx EST ordinal of device dn Device number	Inform site analyst.	PFM
EQxx,DNdn, REPLACE ERROR, AT addr.	The same file was found twice during a catalog search. This error can occur for APPEND or REPLACE commands or macros after a file is found and purged and the catalog search is continued. Error log and dayfile message. EQ Equipment type as defined in the EQxx,DNdn, DIRECT ACCESS FILE ERROR, AT addr. message xx EST ordinal of device dn Device number	Retry job step.	PFM
EQxx,DNdn, SYSTEM SECTOR ERROR, AT addr.	The system sector of an indirect access permanent file contains an error. Error log and dayfile message. EQ Equipment type as defined in EQxx, DNdn, DIRECT ACCESS FILE ERROR, AT addr. message description xx EST ordinal of device dn Device number	Inform site analyst.	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
EQxx,DNdn, TRACK LIMIT, AT addr.	No allocatable tracks remain on equipment xx. Error log and dayfile message. EQ Equipment type as defined in the EQxx,DNdn, DIRECT ACCESS FILE ERROR, AT addr. message xx EST ordinal of device dn Device number	Inform site analyst.	PFM
EQxx,FM=family,PF=filenam,UI=userin.	Additional line is written only in error log after one of the following messages. EQxx,DNdn,DIRECT ACCESS FILE ERROR, AT addr. EQxx,DNdn,FILE LENGTH ERROR, AT addr. EQxx,DNdn,MASS STORAGE ERROR, AT addr. EQxx,DNdn,RANDOM INDEX ERROR, AT addr. EQxx,DNdn,REPLACE ERROR, AT addr. EQxx,Dndn,SYSTEM SECTOR ERROR, AT addr. EQxx,DNdn,TRACK LIMIT, AT addr. EQ Equipment type as defined in the EQxx,DNdn, DIRECT ACCESS FILE ERROR, AT addr. message xx EST ordinal of device family Family name filenam Permanent file name userin User index	None.	PFM
EQxx, nnnn PRINT ERRORS.	Print errors detected on line printer xx. EQ One of the following equipment types. LP Any line printer LR 580-12 line printer LS 580-16 line printer LT 580-20 line printer xx EST ordinal of line printer nnnn Octal number of print errors	Inform customer engineer.	1CD, QAP
EQUIPMENT NOT AVAILABLE.	Tape assignment error was encountered; requested equipment is either in use or is not defined in the system.	Ensure accuracy of macro or control statement and/or retry at a later time.	LFM, RESEX
ERASE LIMIT, lfn AT addr.	The system made 20 erasures (10 feet of tape) without being able to successfully write the tape.	Clean tape or use different tape.	1MT
ERROR AT LINE xxx.	Errors have occurred while resequencing a BASIC program. The line containing the error is specified by xxx.	Examine line xxx of program to determine cause of error.	RESEQ

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ERROR CODE ec, lfn AT addr.	1MT error code ec has occurred but no specific message is issued. This would normally not occur unless the job was dropped by the operator.	Consult site personnel.	1MT
ERROR - FILES(S) NOT PROCESSED.	One or more files were not checkpointed because CHKPT detected address errors.	Determine and correct address errors and retry.	CHKPT
ERROR FLAG TERMINATION, lfn AT addr.	The job was aborted with an input/output request or tape operation in progress. The operation/request is not complete. For example, the operator could kill the job while tape error recovery is in progress.	None.	CIO, 1MT
ERROR IN ARGUMENT.	One of the following. - The pfn is blank or fn = lfn. - The user specified no arguments or a blank argument. - The user specified too many files. - The user entered an illegal parameter.	Reenter the command or control statement with correct parameters.	PFILES
ERROR IN ARGUMENTS.	One or more of the following control statement errors were detected. - More than one date was entered. - No options were selected. - The parameter was illegal or could not be recognized. - The TM option was selected but no data was specified. - Both the device number parameter and the pack name or auxiliary device parameter were selected; auxiliary devices do not have device numbers.	Correct control statement and retry.	BLANK, PURGALL
ERROR IN CONTROL ARGUMENTS.	Either the number of arguments was illegal, an illegal argument was specified, or a numeric parameter was nonnumeric.	Check the description of the control statement.	CONTROL
ERROR IN DATE.	The format of the date (ad, md, or cd) parameter in a PURGALL request was incorrect.	Check the format of the PURGALL statement in section 8.	PURGALL
ERROR IN DEVICE NUMBER.	The file residency as specified by the device number parameter was illegal.	Correct the device number parameter.	PURGALL
ERROR IN FCOPY ARGUMENTS.	The format of the FCOPY control statement is in error.	Refer to the description of the FCOPY control statement.	FCOPY

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ERROR IN FILE ARGUMENTS.	The parameter could not be recognized.	Compare the parameters specified with the control statement description.	FILES
ERROR IN FILE CATEGORY.	The user specified an illegal file category.	Refer to description of file categories for valid entry.	PFILES, PURGALL
ERROR IN FILE DATA, lfn AT addr.	An error was encountered in the data for file lfn.	Inform site analyst.	PFM
ERROR IN FILE TYPE.	The user specified an illegal file type.	Check the file type descriptions in section 2.	PURGALL
ERROR IN LIMITS ARGUMENT.	Dayfile message indicating that parameters were included on the LIMITS statement.	Enter LIMITS. without additional parameters.	MODVAL
ERROR IN PASSWOR ARGUMENTS.	PASSWOR control statement or command parameters are incorrect.	Correct control statement or command and reenter.	PASSWOR, MODVAL
ERROR IN PERMIT DATA, lfn AT addr.	An error was encountered in the permit entries for file lfn.	Inform site analyst.	PFM
ERROR IN PROFILE ARGUMENTS.	Dayfile message indicating there was an error on the PROFILE control statement.	Correct the control statement and rerun.	PROFILE
ERROR IN ROUTE FUNCTION, LFN=filenam.	Informative message issued to the system dayfile stating an error occurred while routing filenam.	None.	DSP
ERROR IN TIME.	The format of the time parameter in a PURGALL request was incorrect.	Check the format of the PURGALL statement in section 8.	PURGALL
ERROR LIMIT EXCEEDED.	Number of parity and block-too-large errors exceed the error limit.	If dayfile shows block-too-large errors have occurred and tape is S, L, or F format, increase block size and retry; otherwise, tape is probably assigned in the wrong format. If parity errors have occurred, the tape is bad and the data on it cannot be correctly recovered.	COPY, TCOPY

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
EXCHANGE PACKAGE/MEMORY DUMP ON FILE ZZZDUMP.	The exchange package and memory dump is written on local file ZZZDUMP because the job is of time-sharing origin and file OUTPUT is assigned to a terminal.	To examine the exchange package and dump, list file ZZZDUMP.	CPMEM
EXU ARG. ERROR.	The address of the arguments is out of range or the program name is zero.	Correct error and retry.	EXU
FCOPY COMPLETE.	The FCOPY conversion is finished.	None.	FCOPY
FILE BOI/EOI/UI MISMATCH, lfn AT addr.	Data in the system sector for file lfn does not match information from the EOI sector and/or catalog information.	Inform site analyst.	PFM
FILE BUSY.	The file specified in the request cannot be interlocked.	Wait until file is not busy.	LFM
FILE EMPTY.	The Update program library contained no data.	None.	UPMOD
FILE EMPTY.	The file specified was empty.	Verify that the file contains data and retry.	LFM, SFM
FILE ERROR filenam.	An illegal address was detected on file filenam.	Correct error and retry.	CHKPT, RESTART
FILE NAME CONFLICT.	Either the user tried to process two files having the same name or he specified a reserved file name.	Correct the control statement so all files have a unique name.	CONVERT, FCOPY, TCOPY, COPY, VERIFY, LIBGEN, UPMOD
FILE NAME CONFLICT.	The S and L file names are the same.	None.	LO72
FILE NAME ERROR.	File name contains illegal characters or contains more than seven characters.	Ensure that legal file name is specified.	LFM, LIBGEN, LISTLB
FILE NAME ERROR, AT addr.	File name contains illegal characters.	Verify that file name contains only valid characters.	PFM
FILE NOT FOUND - filenam.	Warning message that informs the user that the designated file did not exist prior to the copy or verify.	If file should have existed, reaccess the file and retry copy.	COPY, COPYBR, COPYBF, COPYEI, COPYX, TCOPY, VERIFY

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
FILE NOT FOUND.	Requested file could not be found.	Verify that file exists and retry.	LFM, SFM, QFM, ENQUIRE, STIMULA
FILE NOT ON MASS STORAGE.	The specified file does not reside on mass storage.	Copy file to mass storage and retry.	1AJ
FILE STRUCTURES NOT COMPATIBLE.	Results are not guaranteed when the logical structures of the files being verified are not compatible.	To guarantee correct results, verify that files have same logical structure.	VERIFY
FILE TOO LONG.	File cannot be sorted in memory. This message can also indicate that the files were out of order.	Either rerun with the files in the correct order or submit a sort job.	MSORT
FILE TOO LONG, AT addr.	The local file specified for a SAVE, REPLACE, or APPEND command exceeds the length allowed or the direct access file specified for an ATTACH operation in WRITE, MODIFY, or APPEND mode exceeds the direct access file length limit for which the user is validated.	Reduce length of file or save as a direct access file.	PFM
FILENAME CONFLICT.	The first two parameters of the GTR control statement are identical.	Specify different file names for the first two parameters on the GTR statement.	GTR
FL BEYOND MFL (ECS).	ECS field length requirements for the job step exceed the ECS field length allowed.	Increase job step ECS field length.	1MA
FL REQUEST BEYOND MFL (CM).	CM field length requirements for the job step exceed the CM field length allowed.	Increase job step CM field length.	1MA, TCS
FL TOO SHORT FOR PROGRAM.	The user's field length is too short for the program.	Rerun the job with larger field length specification.	TCS, 1AJ
FLE TOO SHORT FOR LOAD.	An attempt was made to load ECS data beyond the user's ECS field length.	None.	1AJ, LDR
FM NOT LEGAL FAMILY.	Dayfile message indicating that an illegal family name was specified with the FM parameter.	Correct the FM parameter and rerun.	PROFILE, MODVAL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
FORMAT ERROR.	One of the following. - The GTR control statement format was incorrect. - An illegal library type was specified. - A record name longer than seven characters was specified.	Refer to the description of the GTR control statement.	GTR
FORMAT ERROR ON CONTROL CARD.	An error was detected in the format of the control statement.	Check the description of control statement format.	TCS
FORMAT REQUIRES UNLABELED TAPE.	The format specified (F) is valid only for unlabeled tapes.	The tape must be assigned as an unlabeled tape.	RESEX
FR INVALID FOR THIS OPTION.	User entered a string parameter with the I option.	Reenter DAYFILE command with correct parameter.	DAYFILE
FUNCTION REJECT, lfn AT addr.	Function was rejected (possible hardware problem).	Inform site analyst.	IMT
FWA .GE. LWA+1.	The first word address parameter was greater than the last word address parameter on DMP, DMD, DMPECS, DMDECS, LOC, or PBC control statements or DMP, DMD, DED, or DEP system requests.	Correct error and retry.	CPMEM
FWA/LWA .GE. FL.	Either first word address parameter of LOC or the last word address of LOC or PBC was greater than or equal to the user's field length.	Reduce FWA and/or LWA and retry.	CPMEM
GTR ERRORS.	GTR detected an error with the call.	Check the description of the GTR statement.	GTR
H VALUE INVALID.	The H parameter was zero or greater than the buffer length.	None.	LO72
HTIME xxxxxxxxxxx.xxx KCYCLES.	Dayfile message giving the CYBER 170 model 176 CPU clock cycle count for the job. The count is in kilocycle units.	None.	IAJ
HTIME NOT AVAILABLE.	The HTIME control statement or macro is valid only on a CYBER 170 model 176.	None.	IAJ
I/O ON EXECUTE-ONLY FILE lfn AT addr.	The user attempted to read, write, or position an execute-only file. RETURN is the only operation allowed for an execute-only file.	None.	CIO

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
I/O SEQUENCE ERROR.	Action was requested on a busy file.	Wait until file is not busy and retry.	CFM, LFM
I/O SEQUENCE ERROR, AT addr.	A request was attempted on a local file that is currently active. This error can occur, for example, if the user creates two FETs for the same file and issues a second request before the first is completed.	Wait until file is inactive.	PFM, LFM
I/O SEQUENCE ERROR ON FILE lfn AT addr.	The user attempted to perform more than one concurrent function on a single file.	Wait until each function is complete before attempting another.	CIO
ILLEGAL CHANGE IN FILE/ORIGIN TYPE.	LFM attempted to change the file/origin type of a deferred routed output file.	Rescind prior routing (DC=SC option).	LFM
ILLEGAL CHARACTER NUMBER.	In a copy request, one of the following was detected. - Last character position was less than first character position. - Last character position was greater than 150. - Either first character position or last character position was unrecognizable.	Ensure accuracy of control statement parameters.	COPYCF, COPYCR
ILLEGAL CHARGE.	Dayfile and output file message indicating one of the following. - The charge or project number does not exist. - The project number is not available to a user with this user number. - The charge or project number exists but is inactive.	Check to see that charge and project numbers are correct and reenter.	CHARGE
ILLEGAL CODE, FILE xxxx, REC yyyy.	In file xxxx, record yyyy of lfn1 (as named in the FCOPY control statement) a code exists which does not have a corresponding code in the character code set of lfn2. FCOPY converts the code to a space and continues converting lfn1.	List lfn1 using the TDUMP control statement to determine the code converted to a space. Refer to appendix A for listings of the character code sets.	FCOPY
ILLEGAL COMBINATION OF QN/SI.	SI was specified and QN was not. If QN is specified, then SI must also be specified.	Specify SI and rerun the job.	LISTLB
ILLEGAL CONTROL CARD.	One of the following has occurred. - The control statement could not be identified. - The USER control statement does not have a user number specified. - An invalid parameter was specified or	Ensure accuracy and/or suitability of control statement.	TCS, CHARGE, CONFIG, MODVAL, RESEX, EXU,

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	no terminator was detected. - The user included too many parameters on the program call statement (such as LGO). - The user submitted a control statement that he was not validated to use (for example, the use of PASSWOR by user not validated to change password). - The user submitted a control statement that is illegal for a particular job type or file type (for example, the use of a FAMILY statement in a nonsystem origin job). - RESEX entry points LFM, PFM, or REQ entered via a control statement.		PASSWOR, 026
ILLEGAL COPY.	File and/or conversion types do not meet copy requirements.	Correct error and retry.	COPY, TCOPY
ILLEGAL COUNT.	The copy file count is nonnumeric or specified as zero.	Correct control statement and retry.	COPYCF, COPYCR, COPYSBF
ILLEGAL DEVICE REQUEST, AT addr.	The device type (r parameter) specified on a request for an auxiliary device cannot be recognized or does not exist in the system. If the auxiliary device specified by the pn parameter is not the same type as the system default, the r parameter must be included; if not, the message is issued.	Examine auxiliary device request and ensure its accuracy.	PFM
ILLEGAL DISPOSE CODE.	The queue type specified on a DISPOSE control statement was unrecognizable.	Consult description of DISPOSE control statement for valid queue types.	FILES
ILLEGAL EQUIPMENT.	Equipment specified does not exist or is not allowed (for example, a TT device is requested from other than terminal origin, or a tape is being requested with the REQUEST macro).	Ensure file resides on a legal equipment type.	LFM, RESEX
ILLEGAL ERROR EXIT ADDRESS.	Error exit address is beyond user's current field length.	Informative.	IAJ
ILLEGAL EXTENSION OF lfn AT addr.	The user attempted to lengthen a file that could not be extended.	Verify that valid file is being extended.	CIO

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ILLEGAL EXTERNAL CALL.	RESEX did not recognize external call.	Inform site analyst.	RESEX
ILLEGAL FILE MODE.	The user tried to dispose or unlock a file which was in execute-only mode or tried to change its file type to library (LIFT).	None.	LFM
ILLEGAL FILE NAME lfn AT addr.	The file name does not conform to established rules.	Use valid file name.	CIO
ILLEGAL FILE TYPE.	The specified file is of a type not allowed in the requested operation. Possible causes include attempts to - change a nonlocal file to file type library - designate a direct access file as the primary file - ROUTE or DISPOSE the primary file	Verify that file type is valid.	LFM
ILLEGAL HOLL. CODE, RECxxx CDyyyy.	Statement yyyy (octal count) in record xxx was found to contain an illegal Hollerith code.	Correct statement in error.	ICD
ILLEGAL I/O REQUEST ON FILE lfn AT addr.	CIO could not recognize the specified function code, or the code was not valid for the type of device to which the file was assigned. The system provides a dump of the FET on file OUTPUT.	Verify CIO function code being used.	CIO
ILLEGAL ID CODE.	An identification code specified on the SETID control statement or macro is not a valid device identification code as defined in the installation EST.	Reissue the request with a valid ID parameter.	LFM
ILLEGAL INPUT FILE.	An attempt was made to pack a file that is assigned to a time-sharing terminal. For example, file INPUT for time-sharing origin jobs cannot be packed.	Verify that file being packed is not assigned to terminal (TT).	PACK
ILLEGAL INSTRUCTION.	The CPU attempted to execute an illegal or nonavailable instruction.	Analyze job output and dumps to determine the cause of the error.	IAJ
ILLEGAL JOB/USER CARDS.	A job was submitted with an invalid job or user card.	Correct job and rerun.	QFM
ILLEGAL LABEL TYPE, lfn AT addr.	Illegal label type. Only legal label types are ANSI labeled and nonstandard labeled.	Use correct label type.	IMT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ILLEGAL LOAD ADDRESS.	The load address is less than 2.	Specify larger load address and retry.	IAJ
ILLEGAL MODIFICATION OF lfn AT addr.	Either the user has attempted to shorten a modify-only file or the file cannot be modified at all.	Determine whether file can be modified.	CIO
ILLEGAL OPTION.	The option specified is not defined for ENQUIRE.	Check parameters on control statement and retry.	ENQUIRE
ILLEGAL PARAMETER.	One of the following. - Parameter value is outside legal bounds. - The user specified a parameter that cannot be included on the command or control statement. - Command/control statement is invalid.	Ensure accuracy of command/control statement.	IAFEX, LISTLB, CATLIST, TELEX
ILLEGAL PASSWORD.	One of the following. - The password entered is greater than seven characters or contains an invalid character. - In the PASSWOR command either an incorrect old password was specified or the new password was unacceptable. - In the MODVAL control statement (for a create or update run) the password for a new user contained fewer characters than the minimum length required by the site. If entered from a K display, the line of input is ignored; otherwise, that particular user number is disregarded.	Correct error and retry.	PASSWOR, MODVAL, PFILES
ILLEGAL QUEUE SPECIFIED.	Queue type specified is illegal.	Retry with valid queue type.	SUBMIT
ILLEGAL REQUEST.	No parameters were specified on a DMPECS or DMDECS control statement.	Retry job with corrected control statement.	CPMEM
ILLEGAL RESOURCE COUNT.	Total resource demand exceeds maximum allowed, as defined by installation parameter MAXD (defined in RESEX).	Reduce RESOURC demands.	RESEX
ILLEGAL SORT PARAMETER.	An illegal parameter has been specified on the SORT control statement.	Consult description of SORT control statement for valid parameters.	SORT
ILLEGAL TERMINAL REQUEST.	A command intended for time-sharing origin jobs only has been used in a nontime-sharing origin job.	None.	IAFEX, TELEX

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ILLEGAL USER ACCESS.	Dayfile message indicating that the user tried to perform an operation for which he is not validated. Possible causes include attempts to - run a system orgin job from nonsystem origin - access a restricted subsystem without proper validation - enter an invalid SRU value - use the V carriage control character without validation	Ensure accuracy of control statement or determine proper validation requirements via LIMITS statement.	LFM, NETVAL, QFSP, RESEX, IMA
ILLEGAL USER ACCESS, AT addr.	The user is not validated to create direct access or indirect access files or to access auxiliary devices.	Contact installation personnel concerning validations.	PFM
ILLEGAL USER ACCESS - CONTACT SITE OPR.	The security count for the user number specified has been decremented to zero. Therefore, the user is denied all access to the operating system until the operator resets the security count.	Inform site operator.	CPM, NETVAL, IAJ, ILS
ILLEGAL USER CARD.	The user attempted to submit a file which does not have a USER statement as the second statement, has an incorrectly formatted USER/ACCOUNT statement, or has an invalid family name, user name, or password.	Correct error and rerun.	QFM
ILLEGAL USER CARD.	The user number or password could not be validated, or a secondary user statement was encountered while secondary user statements were disabled. Dayfile message.	Verify that user number and password are valid. If secondary user statements are disabled, ensure that no secondary user statements are present.	CPM, ACCFAM
ILLEGAL XL BUFFER/FET LABEL FIELD, lfn AT addr.	One of the following. - HDR1 label in extended label buffer or FET contains a nonnumeric display code value in a numeric field. - Character count in header word preceding labels in the extended label buffer does not equal 80.	Correct condition that caused error and retry.	IMT
IMPROPER ACCESSIBILITY.	The user did not specify the correct file accessibility on the LABEL statement or macro, or volume accessibility was set and a nonsystem origin user attempted to assign the tape as unlabeled.	Ensure accuracy of request.	RESEX

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
IMPROPER VALIDATION.	A validation program (one containing a VAL= entry point, such as that used for CHARGE and USER) is required before continuing.	Verify that USER statement precedes rest of job (followed by CHARGE, if required) and is the first statement after the job card.	TCS
INITIAL CONTROL STATEMENT LIMIT.	The number of control statements processed by the job exceeded the limit for which the user is validated. At this point, an additional eight statements are allowed for error processing.	Reduce the number of control statements to a value within the specified limit.	IAJ
INPUT FILE ERROR.	An error on the input file was encountered during the unpack operation.	None.	LO72
INPUT FILE IN NORERUN STATUS.	Informative message.	None.	QFM
INPUT FILE IN RERUN STATUS.	Informative message.	None.	QFM
INQUIRY COMPLETE.	Dayfile message indicating that the inquiry is completed.	None.	MODVAL
INSUFFICIENT RESOURCES ON SYSTEM.	Resource demand exceeds number of units physically available on the system.	Reduce resource demand.	RESEX
INSUFFICIENT STORAGE FOR LIBRARY GENERATION.	Additional memory is required for the LIBGEN control statement.	Increase field length and retry.	LIBGEN
INVALID LINE LENGTH.	Either of the following out of bounds conditions exists with respect to the Ix, Nx, Ox, and H parameters of the LO72 control statement. - (Ox+Nx).GT.H - (Ix+Nx).GT.(buffer length) where 1 .LE. x .LE. b	None.	LO72
INVALID NOISE SIZE ON filenam.	Noise size on S, L, or F format tape does not meet copy requirements.	For copy, ensure that noise size on input file is greater than or equal to that of the output file. For TCOPY, correct noise size to meet requirements and retry (noise size of 8 for 7-track or 6 for 9-track tapes is required for X and SI format conversions).	COPY, TCOPY

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
INVALID PARAMETER.	The S or L parameter was entered as zero on the L072 control statement.	None.	L072
IX OR OX NOT DEFINED.	The Ix or Ox parameter was not spcified in conjunction with the Nx parameter on a L072 control statement.	Check the description of the L072 control statement in section 7.	L072
JOB CARD ERROR (jobcar)	An error was encountered on the job statement in the routed or submitted file. The first 20 characters of the job statement are displayed.	Correct job statement and rerun.	DSP, QFM
JOB CARD ERROR.	The job statement on the file being submitted is in error.	Compare the job statement in error with the job statement description in section 5.	IAJ
JOB COMPLETE. JOB NAME IS jobnam.	Informative message indicating that the job name of the routed file is jobnam.	None.	ROUTE
JOB IN NORERUN STATE ON RECOVERY.	Identifies a job recovered on level 0 deadstart that was aborted because it was in a no-rerun mode (due to NORERUN control statement or macro).	Refer to the NORERUN control statement or macro.	IAJ
JOB REPRIEVED.	The job has been successfully reprieved.	None.	SFP
JOB STEP EXCEEDS ACCOUNT BLOCK.	The user tried to set his job step limit to a value greater than his account block limit or tried to set his account block limit to a value less than his job step limit.	Check values on SETJSL and SETASL statements.	CPM
JOB STEP LIMIT.	The monitor detected the expiration of the job step SRU limit.	Reset job step limit with SETJSL control statement or macro and retry. If job step SRU limit is set at maximum, request increased SRU validation.	IAJ
LABEL CONTENT ERROR, lfn AT addr.	A block read was the correct size for a label but one or more required fields (such as the label name) were incorrect.	Use LISTLB control statement to obtain label data.	IMT
LABEL MISSING, lfn AT addr.	During a read operation, a required label was missing.	Ensure that tape has label.	IMT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
LABEL PARAMETER CONFLICT ON OPEN, lfn AT addr.	Label fields did not match on open request. An additional message FIELD BEGINNING AT addr NO COMPARE. specifying the decimal character position in HDR1 of the first field that did not compare correctly is also issued.	Use LISTLB control statement to obtain label data.	IMT
LDR ERROR.	Issued after one of the following errors. OVERLAY NOT FOUND IN LIBRARY. ARGUMENT ERROR.	Correct error and retry.	LDR
LFM ARG. ERROR.	LFM detected an error in the request.	Ensure that valid LFM request is being made.	LFM
LFM ILLEGAL REQUEST.	One of the following. - The LFM function detected was not recognized as a legal function. - An LFM function was issued without the auto recall bit set.	Verify that valid LFM request is being used.	LFM
LIBGEN ARGUMENT ERROR.	An invalid parameter was used on the LIBGEN control statement.	Check the format of the LIBGEN control statement.	LIBGEN
LIBGEN FILE NAME CONFLICT.	The LIBGEN statement named the same file as the input file and as the output file.	Change the input file or output file name.	LIBGEN
LIBRARY GENERATION COMPLETE.	Message issued when generation of a library is completed.	None.	LIBGEN
LIBRARY GENERATION FILE EMPTY.	The file to be processed is empty.	Verify that file is local to job and contains data.	LIBGEN
LINE NUMBER LIMIT EXCEEDED.	The line number encountered or required during a resequencing (RESEQ) operation exceeded 99999.	Examine program and correct line number in error.	RESEQ
LINE(S) TRUNCATED.	Informative message. File lfn1 contains lines longer than the maximum length processed by the FCOPY. These lines were truncated when written to lfn2.	None.	FCOPY
LISTLB ABORT.	A fatal error has occurred while processing the LISTLB control statement.	Refer to dayfile for cause of problem.	LISTLB
LISTLB COMPLETE.	Informative message indicating that the LISTLB operation has finished.	None.	LISTLB

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
LOCAL FILE LIMIT.	The user has too many local files.	Return one or more local files and retry.	LFM, QFM
LOCAL FILE LIMIT, AT addr.	The job's local file limit has been exceeded in an attempt to GET or ATTACH the file.	Return one or more local files.	PFM
LOCAL FILE LIMIT, FILE lfn AT addr.	The job's local file limit was exceeded in an attempt to define another file or attach an existing file to the job.	Return one or more local files.	CIO
LO72 COMPLETE.	Informative message indicating that the program has completed processing.	None.	LO72
LPxx,....	Refer to description of corresponding message beginning with EQ.		
LRxx,....	Refer to description of corresponding message beginning with EQ.		
LSxx,....	Refer to description of corresponding message beginning with EQ.		
LTxx,....	Refer to description of corresponding message beginning with EQ.		
M.T. NOT AVAILABLE ON FILE lfn AT addr.	The magnetic tape executive is not executing.	Inform site operator.	CIO
MAGNET NOT ACTIVE.	No UDT address or incorrect UDT address in FST or MAGNET not present.	Inform site analyst.	LFM, RESEX
MASS STORAGE DIRECTORY NOT WRITTEN.	On a GTR control statement, the user has requested that a mass storage directory record be written on a nonmass storage file.	Ensure that file resides on mass storage.	GTR
MASTER USER NUMBER REQUIRED.	Dayfile message indicating that the job did not enter user number (via USER statement). This is needed for a master user list run and for a master user inquire run.	Rerun job with USER statement validation.	PROFILE
MESSAGE LIMIT.	The number of messages the job issued has exceeded the limit for which the user is validated. Message functions issued by compilers or applications programs that run at the user's job control point are also counted as user dayfile messages and thus are subject to the user's validated dayfile message limit.	Split job into two or more jobs and retry.	IAJ, IMA

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
MFL LESS THAN ECS MINIMUM CM FL.	To use ECS the user must have a required minimum amount of CM FL. This message indicates user does not have the required CM FL.	Increase CM FL.	IMA
MFL REQUEST TOO SMALL, MINIMUM USED.	MFL request was less than CONTROL's RFL= value. CONTROL's RFL= value is used for this MFL request, thus allowing further MFL requests.	Check the description of field length control in section 3.	CONTROL
MISSING DEMAND FILE ENTRY.	Dayfile message indicating RESEX internal problem. The overcommitment algorithm was initiated without a demand file entry having been defined previously.	Inform site analyst.	RESEX
MISSING VSN OR EQUIPMENT ASSIGNMENT.	Dayfile message indicating internal malfunction in RESEX (expected VSN or equipment assignment was not found).	Inform site analyst.	RESEX
MONITOR CALL ERROR.	RA+1 call unrecognized.	Examine program to determine why illegal RA+1 call is being made.	IAJ
MT,Ccc,Eec,Hhhhhhhhh,B.C.RESTART.	Magnetic tape controller controlware restarted.	None.	IMT
MT,Ccc,Eec,Hhhhhhhhh,BAD ERASE.	Error detected after an erase was attempted to recover a write error.	Inform site analyst.	IMT
MT,Ccc,Eec,Hhhhhhhhh,BID RECOVERY-x.	A single block mispositioning error was recovered by block ID recovery. If x is B, the error was caused by backspacing the tape too far; if x is F, the tape was not backspaced far enough.	None.	IMT
MT,Ccc,Eec,Hhhhhhhhh,BLOCK TOO LARGE.	Data block is at least one byte longer than length bbbb shown in third line of message.	None.	IMT
MT,Ccc,Eec,Hhhhhhhhh,BUSY.	Unit was still busy after 1 second.	Inform customer engineer.	IMT
MT,Ccc,Eec,Hhhhhhhhh,CHANNEL ILL.	Channel is not accepting function for status requests properly.	Inform customer engineer.	IMT
MT,Ccc,Eec,Hhhhhhhhh,CON.REJ.	Connect reject; unable to connect to the unit.	Inform site analyst.	IMT
MT,Ccc,Eec,Hhhhhhhhh,CON REJ. MDI.	Connect reject; unable to connect to unit because of marginal detection indication (thermal warning). Unit turned off.	Inform customer engineer.	IMT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
MT,Ccc,Eec,Hhhhhhhhh,CON.REJ.OFF.	Connect reject; unable to connect to unit. Unit turned off.	Inform site analyst.	IMT
MT,Ccc,Eec,Hhhhhhhhh,FNff,Pyyyy.	Function ff was rejected by the controller; yyyy is the address in IMT where the function was initiated.	Inform site analyst.	IMT
MT,Ccc,Eec,Hhhhhhhhh,Lbbbb,Bnnnnnn.	The length (bbbb) and block number (nnnnnn) read from trailer bytes in block did not match the actual length or the block number read; given in previous message line.	None.	IMT
MT,Ccc,Eec,Hhhhhhhhh,LOAD CHECK.	Load sequence failed on the unit.	Push CLEAR button and reload tape or contact site analyst.	IMT
MT,Ccc,Eec,Hhhhhhhhh,MARGINAL DOWN.	Indicates controller failure. Channel has been logically turned off and maintenance is required.	Inform customer engineer.	IMT
MT,Ccc,Eec,Hhhhhhhhh,MARGINAL OFF.	Unit has been logically turned off because of read/write failure. This occurred when a special function to check the read/write path to a unit failed during initial label scan. Maintenance is required.	Inform customer engineer.	IMT
MT,Ccc,Eec,Hhhhhhhhh,NO EOP.	No end-of-operation detected from unit within 1 second.	Inform customer engineer.	IMT
MT,Ccc,Eec,Hhhhhhhhh,NOISE.	A noise block was skipped on the tape.	None.	IMT
MT,Ccc,Eec,Hhhhhhhhh,NOT READY.	Tape unit dropped ready status.	Make unit ready.	IMT
MT,Ccc,Eec,Hhhhhhhhh,ON THE FLY.	Error was corrected as the data was read.	None.	IMT
MT,Ccc,Eec,Hhhhhhhhh,POSITION LOST.	The last good block written cannot be found during write recovery.	None.	IMT
MT,Ccc,Eec,Hhhhhhhhh,RECOVERED.	Previously reported error has been successfully recovered.	None.	IMT
MT,Ccc,Eec,Hhhhhhhhh,STATUS.	Error type cannot be determined so actual controller status is returned.	Inform site analyst.	IMT
MT,Ccc,Eec,Hhhhhhhhh,WRONG PARITY.	Tape was written in parity opposite that being read.	None.	IMT
MT,Ccc-e-uu,vsn,rw,xx,Ss,GSgggggggg. MT,Ccc,Dddd...d. MT,Ccc,Uuu...u,Ttttt. MT,Ccc,Fff,Iii,Bnnnnnn,Lbbbb,Pppppppppp.	Five-line message describing a magnetic tape hardware malfunction on a 67x tape unit. Message as illustrated indicates 7-track, model 677 unit. If NT appears in	Refer to action indicated opposite individual fifth line message listed according	IMT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
MT,Ccc,Eec,Hhhhhhhhh,type.	place of MT, message indicates 9-track, model 679 unit. Message is issued to error log and dayfile. The first line of the message provides the following information. cc-e-uu — Channel, equipment (tape controller), and physical unit number of tape unit on which error was encountered. vsn — Volume serial number associated with tape on the specified unit. rw — Read (RD) or write (WR) operation; any operation not involving an actual read or write is listed as a read. xx — EST ordinal of the unit on which the tape was written. This is provided only for labeled tapes generated under NOS; otherwise, the field is blank. s — Channel status. gggggggg — General status of magnetic tape unit. Last byte is block ID. The second line of the message provides the following information. cc — Channel number; the channel number is repeated to allow the analyst to associate this message with the first message if errors are occurring on more than one tape channel at the same time. ddd...d — Detailed status of magnetic tape unit. The third line of the message provides the following information. cc — Channel number; repeated to associate this message with the previous messages. uu...u — Unit status of the magnetic tape unit. tttt — Third byte of the tape unit format parameters (refer to the magnetic tape subsystem reference manual for descriptions of unit format parameter fields).	to type field.	

write is listed as a read.

xx — EST ordinal of the unit on which the tape was written. This is provided only for labeled tapes generated under NOS; otherwise, the field is blank.

s — Channel status.

ssssssss — General status of magnetic tape unit. Last byte is block ID.

The second line of the message provides the following information.

cc — Channel number; the channel number is repeated to allow the analyst to associate this message with the first message if errors are occurring on more than one tape channel at the same time.

ddd...d — Detailed status of magnetic tape unit.

The third line of the message provides the following information.

cc — Channel number; repeated to associate this message with the previous messages.

ff — Software function on which the error occurred.

ii — Error iteration; number of times error has been encountered on this unit without successful recovery.

nnnnnn — Block number on which error occurred

bbbb — Length of block on which error occurred, in octal bytes.

pppppppp — IMT internal error parameters.

The fourth line of the message provides the following information.

cc — Channel number; repeated to associate this message with the previous messages.

ec — Octal error code value.

hhhhhhhh — Unit format parameters (refer to the magnetic tape subsystem reference manual for descriptions of unit format

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	parameter fields). type Additional description of the error. Refer to individual fourth-line message.		
MT/NT CONFLICT.	Conflict exists between 7-track and 9-track tape descriptors (track type, density, and conversion mode). For example, a request for a 9-track tape specified 200-bpi density. This message can also be issued if the device type specified in FET+1 conflicts with the track type specified in FET+8, bit 56. If dt=MT and bit 56 is set, the message is issued.	Ensure accuracy of control statement.	RESEX, BLANK
MULTI-FILE NOT FOUND, lfn AT addr.	Either LISTLB has reached the end of the multifile set or the requested file was not found. The following additional messages are also given. REQUESTED SECTION xxxx. FOUND SECTION yyyy. or FILE IDENTIFIER NOT FOUND. The lfn, addr, xxxx, and yyyy given can be ignored.	Ensure that correct tape is being used and that it contains desired file. All label parameters must match in order to position to the specified file.	IMT
NO CONNECT TIME AVAILABLE.	Dayfile message indicating that the user has accumulated the maximum connect time allowed for the specified project number.	Contact installation personnel in order to increase maximum connect time allowed.	CHARGE
NO CPU TIME AVAILABLE.	Dayfile message indicating that the user has accumulated the maximum CPU time allowed for the specified project number.	Contact installation personnel in order to increase maximum CPU time allowed.	CHARGE
NO ECS.	A DMPECS or DMDECS control statement or DED or DEP request was entered and no ECS field length is assigned to the user.	None.	CPMEM
NO FILE(S) RELEASED.	No files of the type PRINT or PUNCH were found local to the user's control point.	Examine control statement record to determine if PRINT or PUNCH files were expected.	OUT
NO HDR1 LABEL RETURNED ON OPEN.	No HDR1 label was found in the label buffer after the OPEN function was completed. Indicates a possible system error.	Inform site analyst.	LISTLB

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
NO INPUT FILE FOUND.	No valid input file exists; functions cannot be performed.	Verify that input file is valid.	QFM
NO LINE NUMBER, FN=filenam.	Input line in file filenam does not contain a line number.	Correct and rerun.	MSORT
NO LINE NUMBER ON SORT FILE.	A line on the input file to a SORT request is missing a line number or a line exceeded the 150-character limit.	Check the format of the input file.	SORT
NO LINE TERMINATOR.	A copy operation was attempted on a line longer than 150 characters which did not contain a line terminator.	Check the format of the file from which the copy was to be made.	COPYCF, COPYCR, COPYSBF
NO MASS STORAGE AVAILABLE.	User attempted to reserve a track for a null primary file.	Retry later.	LFM
NO READ FILE FOUND - filenam.	The file specified on the /READ directive cannot be found.	Ensure that file name specified is correct and that the file is a local file or a permanent file.	SUBMIT
NO SOURCE FILE SPECIFIED.	No file name was specified on the control statement call.	Perform LISTLB to determine expiration date.	SUBMIT
NO WRITE ENABLE, ON lfn AT addr.	Either the user attempted to write on a tape mounted with no write ring or no write was allowed because of additional constraints described in an additional message line. - LABEL NOT EXPIRED. The user attempted to write over a label that had not yet expired. - WRITE OVER LABEL ILLEGAL. The user is not allowed to destroy the VOL1 label. - 200 BPI WRITE ILLEGAL. The tape unit (667 or 677 does not support 200-bpi density.	- Wait. - None. - Specify different tape density.	1MT
NON-MATCHING CONVERSION.	Informative message indicating conversion mode on labeled 9-track tape differs from that specified by assignment request. System writes tape in specified mode, or reads tape with write ring out in correct mode. However, reading tape with write ring in or using wrong conversion mode generates conversion errors.	If reading tape with write ring in, return and reassign with correct conversion mode.	RESEX

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
NON-MATCHING DENSITY.	Informative message indicating that the density specified on the control statement or macro is not the same as the density of the assigned tape. Issued only to 9-track tapes with write ring out. 9-track tapes are read at the current density on tape. They are written at specified density if write initiated from load point; otherwise, tape is written at the current density on the tape.	None.	RESEX
NORERUN/RERUN IGNORED FROM TTY JOBS.	User entered NORERUN or RERUN from a terminal. The command is ignored.	None.	QFM, CONTROL
NT....	Refer to description of MT.... series of messages.		
NT DENSITY CONFLICT.	9-track tape unit specified by EST ordinal on ASSIGN statement does not support the required density.	Ensure that density compatible tape unit is specified.	RESEX
NT DRIVE CONFLICT.	One of the following. 9-track tape unit specified by EST ordinal on ASSIGN statement conflicts with other resource requirements for this job. In this case the system rejects assignment to prevent the job from deadlocking itself. or Increased resource demands (RESOURC control statement) cannot be satisfied due to conflicts with currently assigned resources (job would deadlock itself).	Check the description of the RESOURC statement in section 6. Reduce resource demand which causes conflict.	RESEX
OLDPL ERROR.	Update program library format contains an error.	Correct error and rerun.	UPMOD
OPERATOR DROP.	Informative message indicating that operator dropped job.	None.	DSD, IAJ
OPERATOR EVICT.	The operator entered an EVICT command to drop the job from the rollout queue. This disallows EXIT, EREXIT, and REPRIEVE processing.	Correct job as needed and rerun.	IRI
OPERATOR KILL.	The operator entered a KILL command to drop the job. This disallows EREXIT processing. A job with extended REPRIEVE processing is reprieved once. EXIT processing is allowed.	Correct job as needed and rerun.	IAJ

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
OPERATOR OVERRIDE.	The operator entered an OVERRIDE command to drop the job. This disallows EXIT, EREXIT, and REPRIEVE processing.	Correct job as needed and rerun.	IAJ
OUTPUT FILE LIMIT.	The total number of files disposed to the output queue by the job has exceeded the limit for which the user is validated.	If possible, split job into two or more jobs and retry. Otherwise, reduce number of files by copying output to single file and then issuing dispose.	LFM
OUTPUT FILE LIMIT, FILE lfn AT addr.	During an attempt to close this file, the number of files disposed to output queues by the job has exceeded the limit for which the user is validated.	If possible, split job into two or more jobs and retry. Otherwise, reduce number of files by copying output to single file and then issuing dispose.	CIO
OVERLAY FILE EMPTY.	No data appears in the requested file.	Verify that overlay file is valid.	IAJ
OVERLAY FILE NOT FOUND.	The specified file was not available.	Verify that file is local to job and retry.	IAJ
OVERLAY LOST.	The specified overlay was not found.	Verify that file with specified overlay is local to job.	IAJ
OVERLAY NOT FOUND IN LIBRARY.	The specified overlay was not found in the system library.	Verify that call is to valid overlay.	LDR
PACK PARAMETER ERROR.	The PACK control statement contains too many or no file names.	Check the format of the PACK control statement in section 7.	PACK
PARITY ERROR - RESTARTED FROM kk.	Because RESTART detected a parity error in attempting to restart from the specified checkpoint nn, the alternate checkpoint kk was used instead.	None.	RESTART
PF UTILITY ACTIVE, AT addr.	Because a permanent file utility is currently active, the operation was not attempted; the user should retry the operation.	Wait until PF utility is not active and retry.	PFM
PFM ABORTED, AT addr.	Error flag detected at PFM control point.	Inform site analyst.	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
PFM ILLEGAL REQUEST, AT addr.	One of the following. - Illegal command code passed to PFM - Illegal permit mode or catalog type specified - CATLIST request has permit specified without a file name - PERMIT command or macro attempted on a public file	Verify that PFM request is valid.	PFM
POSITION ERROR ON-filenam.	File filenam was not repositioned after being checkpointed because CHKPT detected an address error.	None.	CHKPT
POSITION LOST, lfn AT addr.	During write or read error recovery, the system could not find the last good block of data, making it impossible to successfully perform error recovery. Labels are not written after this error and existing data on the tape is not destroyed.	Retry operation on different tape and/or tape drive, if possible.	IMT
PP CALL ERROR.	The monitor detected an error in a CPU request for PP action.	Verify that correct PP call is issued.	IAJ
PRIOR TAPE ASSIGNMENT LOST.	Magnetic tape executive has been dropped along with tapes assigned. All of the user's prior tape assignments are lost.	Terminal user must return/unload all prior tapes and reassign. Batch user is aborted and must rerun his job.	RESEX
PROCEDURE FILE EMPTY.	Procedure file specified contains no data.	Verify that procedure file contains data and retry.	CONTROL
PROFILE FILE CREATE COMPLETE.	Dayfile message indicating that the creation run is complete.	None.	PROFILE
PROFILE FILE DATA BASE ERROR.	Dayfile message indicating that the project file does not contain both a level 0 and level 1 block.	Ensure that the project file is local and contains a level 0 and level 1 block (at least one charge entry) and rerun.	PROFILE
PROFILE FILE INQUIRY COMPLETE.	Dayfile message indicating that the inquire run is complete.	None.	PROFILE
PROFILE FILE LIST COMPLETE.	Dayfile message indicating that the list of PROFILa is complete.	None.	PROFILE

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
PROFILE FILE REFORMAT COMPLETE.	Dayfile message indicating that the reformat run is complete.	None.	PROFILE
PROFILE FILE SOURCE COMPLETE.	Dayfile message indicating that the source run is complete.	None.	PROFILE
PROFILE FILE UPDATE COMPLETE.	Dayfile message indicating that the update run is complete.	None.	PROFILE
PROGRAM NOT FOUND.	The program to be loaded was not found on the specified library file.	Verify that the program is on library file.	EXU
PROGRAM NOT ON MASS STORAGE.	The program does not reside on a mass storage device.	Copy program to mass storage.	EXU
PROGRAM STOP.	The system processed a program stop (00) instruction.	None.	1AJ
PROGRAM STOP AT addr.	The monitor detected a program stop instruction at address addr.	None.	1AJ
PROGRAM TOO LONG.	The program does not fit in the available storage.	Increase field length and retry.	EXU
PROJECT NUMBER EXPIRED.	Dayfile and output file message indicating that the project number expiration date has occurred.	None.	CHARGE
PROTECTED FILE.	The user has attempted to release a locked file.	None.	LFM
PRU LIMIT, AT addr.	The job's mass storage PRU limit was exceeded during preparation of a local copy of an indirect access file.	Return one or more local files and retry.	PFM
PRU LIMIT, FILE lfn AT addr.	The job's mass stoage PRU limit was exceeded during an attempt to write or extend this file.	Return one or more local files and retry.	CIO
PRUS REQUESTED UNAVAILABLE, AT addr.	The number of PRUs specified via the S parameter on the DEFINE request is not available.	Request smaller number of PRUs.	PFM
QAP - BUFFER ARGUMENT ERROR.	Dayfile message indicating one of the following. - Buffer does not contain the required number of words of data for the particular function. - Buffer pointer (FIRST, IN, OUT, LIMIT) is out of range.	Verify that the calling program has set up the request correctly.	QAP

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
QAP - ILLEGAL USER ACCESS.	The user tried to perform an operation for which he is not validated (for example, attempting to run a system origin job from nonsystem origin).	Ensure accuracy of control statement or determine proper validation requirements.	QAP
QFM FILE NAME ERROR.	The lfn specified is not a valid file name.	Verify file name.	QFM
QFM FILE NOT FOUND.	The file to be submitted could not be found.	Verify that submit file exists.	QFM
QFM FILE NOT ON MASS STORAGE.	The file to be submitted does not reside on mass storage.	Copy file to be submitted to mass storage and retry.	QFM
QFM ILLEGAL FILE TYPE.	The file to be submitted is not a local file type.	Copy the submit file to a local file.	QFM
RA.SSC OUT OF RANGE.	The subsystem receiving the buffer pointer (RA.SSC) word has invalid data fields.	Correct RA.SSC word fields.	1MA, 1MB
READ AFTER WRITE, lfn AT addr.	The user attempted to read a tape on which the last operation was a write.	Ensure accuracy of tape positioning statements (BKSP, BKSPRU, SKIPFB, or REWIND required to read after write).	1MT
READ FILE BUSY - filenam.	The read file is found to be busy (direct access file only).	Retry after file is not busy.	SUBMIT
READY DROP, lfn AT addr.	Unit dropped ready.	None.	1MT
RECORD SIZE EXCEEDS 500.	The maximum line length for a record to be converted (500 characters) was exceeded.	Split lines that are too long into two or more lines.	CONVERT
RECORD TOO LARGE ON filenam.	An input record was encountered that exceeded S or L output tape block size.	Reduce input record size, or use COPY control statement to increase S or L tape block size or allow record splitting and retry with PO parameter.	COPY, COPYBF, COPYEI
RECORD TOO LONG.	The record is too long for available memory. In response to a WBR request, the record length parameter was greater than or equal to the user's field length. Available memory is filled and the excess data is skipped.	Increase field length and rerun.	CPMEM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
REMOVABLE PACKS OVERCOMMITMENT.	Removable pack request without NA selected causes resource overcommitment.	Retry later or retry with NA parameter on ATTACH, DEFINE, etc., with PN parameter specified.	RESEX
RENAME OF PROC TYPE NOT ALLOWED.	A RENAME of a PROC type record was attempted; this is not allowed.	None.	LIBEDIT
REPRIEVE BLOCK ERROR.	An address is out of range or there is an illegal parameter in the reprieve parameter block at the time of an error. The message is also issued if the specified reprieve address itself is out of range. (1AJ issues this message for all errors except terminal interrupts processed by 1RI.)	Ensure parameter block is correct.	1AJ, 1RI
REPRIEVE CHECKSUM BAD.	The computed checksum does not agree with the checksum specified in the parameter block at the time of the error. (1AJ issues this message for all errors except terminal interrupts processed by 1RI.)	Ensure interrupt handler is still intact. Ensure that code in the area for which checksum was computed has not changed.	1AJ, 1RI
REQUEST UNDEFINED ON DEVICE lfn AT addr.	The specified function cannot be performed on the device on which the file resides. The system provides a dump of the FET on file OUTPUT.	Verify that valid device is specified.	CIO
RERUN NOT POSSIBLE.	The job cannot be rerun because of one of the following. - Job is time-sharing origin. - No input file is found for the job. - An error occurred in reading or writing the input file system sector. - Rerun status is disabled.	None.	1DS
RESEQ CONTROL CARD ERROR.	The RESEQ control statement contains a syntax error.	Correct error and rerun.	RESEQ
RESEQ ERRORS.	A resequencing error was encountered.	Refer to preceding message for more specific information about the error.	RESEQ
RESEQ NUMERIC PARAM ERROR.	A parameter which is supposed to be numeric contains a nonnumeric character.	Correct error and rerun.	RESEQ

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
RESERVED FILE NAME.	A reserved file name was incorrectly used.	Choose a nonreserved file name.	MODIFY, OPLEDIT, EDIT, DATADEF, DAYFILE, IAFEX, TELEX
RESEX ABORT - OPERATOR TERMINATION.	The operator entered a DROP, KILL or RERUN command, setting an error flag in RESEX. RESEX performed appropriate cleanup procedures before termination.	Determine reason for operator action. Rerun job, if desired.	RESEX
RESEX ABORT - SYSTEM RESOURCE LIMIT.	RESEX terminated prematurely due to job time limit, SRU limit, or track limit. RESEX performs appropriate cleanup procedures before termination.	If error caused by SRU or time limit, increase resource limits. If caused by track limit, contact site analyst.	RESEX
RESEX ABORT - TERMINAL INTERRUPT.	Terminal user interrupted RESEX (interrupt or terminate sequence). RESEX performs appropriate cleanup procedures before termination.	None.	RESEX
RESEX DETECTED ERROR.	The resource executive (RESEX) detected an error.	Inform site analyst.	LFM
RESEX FAILURE, AT addr.	The resource executive (RESEX) has detected a fatal error.	Inform site analyst.	PFM
RESOURCE DEMAND ERROR.	The user attempted to decrease the number of scheduled units to less than the number of currently assigned units or increase the number of scheduled units to a point where a deadlock would occur.	Adjust RESOURC statement parameters accordingly.	RESEX
RESOURCE ENVIRONMENT ERROR.	Dayfile message indicating RESEX internal problem occurred (internal environment building failed due to MST, UDT, or EST errors).	Inform site analyst.	RESEX
RESOURCE NEGATIVE SHARE COUNT.	Dayfile message indicating RESEX internal problem occurred. The resource overcommitment algorithm indicates a greater number of users are supposedly sharing a removable pack than are actually sharing the pack.	Inform site analyst.	RESEX

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
RESOURCE PF ERROR ec filenam.	PFM error ec occurred when attaching resource file filenam.	Inform site analyst.	RESEX
RESOURCE SCRATCH FILE ERROR.	Dayfile message indicating RESEX internal problem has occurred. An empty entry has been found in the overcommitment algorithm scratch file.	Inform site analyst.	RESEX
RESOURCE TYPE ERROR.	The user specified an illegal resource type.	Ensure accuracy of control statement.	RESEX
RFL BEYOND MFL.	The RFL request is greater than the maximum field length for a job step.	Increase maximum field length with MFL statement.	CPM
ROLLIN FILE BAD.	A job could not be rolled in correctly.	Inform site analyst. Check error log dayfile for the job that was aborted and the location of the bad rollin file.	IRI
ROUTE COMPLETE.	Informative message indicating ROUTE operation has completed.	None.	ROUTE
ROUTE COMPLETE. JOB NAME IS jobnam.	A ROUTE statement entered a file in the output queue. The routed job has the system job name jobnam.	None.	ROUTE
ROUTE CONTROL CARD ERROR.	Format of the control statement is incorrect.	Check the ROUTE statement format in section 7.	ROUTE
ROUTE *DC* INCOMPATIBLE WITH *EC*.	The user specified a DC/EC combination that is not legal.	If the DC parameter implies a print or punch file, the EC parameter must be for the same file type.	ROUTE
ROUTE *FID* IGNORED.	Informative message listed for NOS/BE compatibility.	None.	ROUTE
ROUTE ILLEGAL KEYWORD.	Control statement contains an illegal keyword.	Check the ROUTE statement format in section 7.	ROUTE
ROUTE ILLEGAL *OT* PARAMETER.	The origin type specified by the OT parameter is illegal.	Verify that correct origin type is specified.	ROUTE

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ROUTE *OT* NOT ALLOWED.	The user program is not system origin. Only system origin jobs can use the OT parameter.	The user without system origin privileges should not attempt to use the OT parameter.	ROUTE
ROUTE *PRI* IGNORED.	Informative message listed for NOS/BE compatibility.	None.	ROUTE
ROUTE *REP* GT 31. DEFAULT USED.	The repeat count specified was greater than 31; it has been set to 0. This condition does not abort the program.	Use a repeat count less than 32.	ROUTE
ROUTE *ST* IGNORED.	Informative message listed for NOS/BE compatibility.	None.	ROUTE
ROUTE *TID* AND *FM/UN* CONFLICT.	The TID parameter was specified with either the FM or UN parameter. Either one of these parameters is mutually exclusive with TID.	Refer to section 7 for the correct format of the ROUTE control statement.	ROUTE
ROUTE *TID/FM/UN* and *ID* CONFLICT.	The ID parameter was specified with the TID, FM, or UN parameter.	Refer to section 7 for the correct format of the ROUTE control statement.	ROUTE
ROUTE *TID=xx. VALUE IGNORED.	Informative message listed for NOS/BE compatibility.	None.	ROUTE
RTIME nnnnnn.nnn SECS.	Dayfile message output by RTIME control statement giving the real-time seconds count.	None.	IAJ
SECURE MEMORY, DUMP DISABLED.	An attempt was made to dump memory protected by the system.	If possible, allocate additional SRUs with S,nnnn command.	IAJ
SFP CALL ERROR.	Dayfile message indicating that SFP was called directly and not by PLL.	Inform site analyst.	SFP
SFP/DOO BAD ERROR TEXT.	The specified systems text is not in the correct format.	Rebuild systems text using correct format.	SFP
SFP/DOO ERROR TEXT NOT FOUND.	The specified systems text could not be found.	Ensure the systems text specified is correctly installed on the system or call site analyst.	SFP
SFP/DOO ERROR TEXT NOT ON MASS STORAGE.	The specified systems text does not reside on mass storage.	Move systems text to mass storage.	SFP

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
S ?/D00 INVALID MESSAGE NUMBER.	The specified message number could not be found on the systems text.	Ensure that D00 was called with the correct message number.	SFP
SFP/xxx ILLEGAL FUNCTION.	The function code in the RA+1 call is in error.	Correct and retry.	SFP
SFP/xxx ILLEGAL FUNCTION CODE.	The specified function code to the program xxx is not defined.	Ensure the correct function code was used.	SFP
SFP/xxx ILLEGAL ORIGIN CODE.	Dayfile message indicating that the function was illegal for the user's origin.	Inform site analyst.	SFP
SFP/xxx PARAMETER ERROR.	Dayfile message indicating that the parameter address was outside the field length.	Inform site analyst.	SFP
SFP/PFE I/O SEQUENCE ERROR.	Action was requested on a file that was already busy.	Wait until the file is not busy and retry.	SFP
SFP/RPV UNABLE TO RESET, NOT REPRIEVED.	Dayfile message indicating that an attempt was made to reset when the job had not been reprieved.	Inform site analyst.	SFP
SFP/SRP SPECIAL REQUEST PROCESSING ERROR.	Dayfile message indicating that the SPCW word was busy.	Inform site analyst.	SFP
SFP/STS UNKNOWN DEVICE TYPE/NAME.	The device code is not recognized by the system.	Check device code for validity.	SFP
SHARE TABLE ERROR.	Dayfile message indicating RESEX internal problem occurred (share table full or matching entry not found).	Inform site analyst.	RESEX
SL NOT VALIDATED.	The SRU limit requested exceeds that for which the user is validated.	Request smaller SRU limit.	CPM
SPCW CALL ERROR.	A DMP= type call was made, and the program called is either not in the CLD or does not have a DMP= entry point defined.	Inform site analyst.	IAJ
STATUS ERROR, lfn AT addr.	An irrecoverable error was encountered. A second message line describes the error in more detail.	Retry or inform site analyst.	IMT
	ARA BURST DEFECTIVE. First block of tape at 1600 or 6250 cpi cannot be read or written. Another unit or tape should be tried.		

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	CRC ERROR. An error was detected in cyclic redundancy character.		
	FILL STATUS ILLEGAL. The system has detected an odd number of frame, a condition which is illegal for the data format of the tape being read.		
	MULTI-TRACK PHASE ERROR. Multiple tracks were found to be in error at 1600 cpi, making recovery impossible.		
	PARITY ERROR. The tape could not be read/written correctly.		
	UNIT HAS MOTION PROBLEMS. The tape unit cannot properly write the tape. The user should resubmit his job, using a different tape unit.		
	UNIT PROBLEMS. Unit check bit is set in detailed status (67x units only). The user should try another unit.		
	POSTAMBLE ERROR. A missing or defective postamble was detected at 1600 cpi.		
	SINGLE FRAME ERROR. A frame (NRZI only) containing all zeros was read; data will be at least one frame short.		
	LRC ERROR. The longitudinal redundancy check character was read incorrectly (9-track NRZI).		
	ILLEGAL CHARACTER. Illegal character read from 9-track tape. If a 1 is detected in bit 6 of a translated character, the character is illegal.		
STEP CONDITION.	Step mode flag set in the PSD register caused the program to interrupt at the end of a program instruction with an exchange jump to EEA (the error exit address in the exchange package).	Inform site analyst.	IAJ

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
SUBMIT FILE EMPTY.	An EOR or EOF was encountered on the submit file before any data was found.	Rewind submit file, verify format of data and that the file is a local file, and retry operation.	SUBMIT
SUBSYSTEM ABORTED.	The user job was connected (either long term connection or wait response set) to a subsystem which aborted.	Retry later.	1AJ
SYSTEM ABORT.	Possible errors include detection of a bad rollout file by 1RI, an unrecognizable error flag, an SSJ= block outside a field length, an invalid USER statement, or an attempt to access a restricted subsystem. Message also issued because operator evicted job.	If a preceding message gave the cause of the abort, refer to the action listed for that message; otherwise, inform site analyst.	1AJ
SYSTEM CHECKPOINT ABORT.	The checkpoint system was up and aborted, and in the process aborted a number of subsystems.	Informative message.	1CK
SYSTEM ERROR.	One of the following. - RESEX cannot communicate wtih subsystem via the RSB or SIC monitor calls. - LFM cannot complete the requested LFM function because the calling program has a DMP= entry point. - BLANK is unable to read low core pointers via the RSB monitor call. - PFM detects error condition.	Inform site analyst.	BLANK, LFM, PFM, RESEX
TABLE OVERFLOW.	The job field length is too small to hold the tables for processing the GTR statement.	Increase field length and rerun job.	GTR
TAPE BLANK LABELED.	Dayfile message indicating that the blank label operation successfully completed.	None.	BLANK
TAPE BLOCK DEFINITION ERROR.	The user attempted to define data block size via the FC or C keyword or noise block size via the NS keyword in such a manner that the system is unable to correctly define the size of the data block. The omission of the FC or C parameter on a control statement where it is required also causes this message to be issued.	Ensure accuracy of control statement.	RESEX
TAPE FORMAT PROBABLY WRONG.	This message is issued in addition to one of the following messages. BLOCK SEQUENCE ERROR, lfn AT addr. BLOCK TOO LARGE, lfn AT addr. WRONG PARITY, lfn AT addr.	Ensure accuracy of format (F) parameter on control statement or macro.	1MT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
TIME LIMIT ENTER T TO CONTINUE OR CR KEY TO STOP:	The monitor detected that the time limit for the job step has expired.	To continue reset time limit with T,nnnnn command or enter T alone to set time limit to an installation-defined default.	IAJ
TL NOT VALIDATED.	The time limit requested exceeds that for which the user is validated.	Request smaller time limit.	CPM, ACCFAM
TOO MANY ARGUMENTS.	The number of arguments on the control statement exceeds that allowed by the program.	Reformat the control statement with an allowable number of arguments.	TCS
TOO MANY DEFERRED BATCH JOBS.	The user is not validated for this function or he has more jobs in the system than he is allowed. (All jobs in local and remote batch queues are counted.) The count is ignored if the job is of system origin or the user is validated for system privileges and DEBUG mode is set by the operator.	If possible, split job into two or more jobs and retry. Otherwise, reduce number of files by copying output to single file and then disposing file.	QFM
TOO MANY FILES - NOT ALL PROCESSED.	The job had more files than could be processed.	If message resulted from a REWIND operation, issue a series of REWIND statements or commands. Otherwise, reenter the statement or command. Inform the site analyst if this error occurs frequently.	MFILES
TOO MANY PARAMETERS.	More parameters were entered (including null parameters) than are allowed for command.	Ensure accuracy of entry.	GTR, IAFEX, TELEX,
TOTAL ASSIGNED COUNT ERROR.	Dayfile message indicating RESEX internal problem (sum of individual resource assigned counts differs from total assigned count in demand file entry).	Inform site analyst.	RESEX
TOTAL DEMAND COUNT ERROR.	Dayfile message indicating RESEX internal problem occurred (sum of individual resource demand counts differs from total demand count in demand file entry).	Inform site analyst.	RESEX

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
TRACK LIMIT, FILE lfn AT addr.	The device on file resides is full.	Specify another device or contact site analyst.	CIO
UNABLE TO READ IQFT FILE.	An attempt to initialize inactive queues failed because the IQFT file could not be read.	Retry operation.	IMS, MSI
UNIDENTIFIED PROGRAM FORMAT.	The file the user requested to be loaded was in an unrecognizable format.	Check the format of the file.	1AJ
UNIT HUNG UP ON EOP OR BUSY, lfn AT addr.	Unit did not receive EOP on unit busy.	Inform site analyst.	1MT
UNLABELED TAPE REQUIRED - filenam.	An S tape used for E, B, or X tape conversion must be unlabeled.	Use unlabeled tape.	TCOPY
UNRECOGNIZABLE TYPE SPECIFIED.	The type of the source file was not legal.	Check type of source file.	LO72
UNRECOVERABLE ERROR ON filenam.	An unrecoverable error such as wrong parity, density change, or ready drop was detected on the input file.	Either the tape was bad or the tape drive is malfunctioning.	COPY, TCOPY
UNUSUAL END-OF-FILE ENCOUNTERED.	GTR detected an EOF mark not preceded by an EOR mark.	Dump file to determine file format error.	GTR
UPDATE COMPLETE.	The user's password has been changed.	None.	PASSWOR
UPMOD COMPLETE.	Message indicating UPMOD completion.	None.	UPMOD
USER NOT VALIDATED FOR ECS MINIMUM CM.	The user was validated for the ECS specified on the job statement but was not validated for at least 10000 octal words of memory.	None.	ACCFAM
VERIFY ERRORS.	Verification errors were encountered during comparison of date/records/files.	Check output file for errors.	VERIFY
VERIFY FL ABOVE USER LIMIT.	Field length for L or F tape verify exceeds user's current maximum.	Use MFL to increase maximum field length or reduce block size on L or F tapes.	VERIFY
VSN FILE ERROR.	Dayfle message indicating RESEX internal problem occurred (VSN file entry does not match job identification).	Inform site analyst.	RESEX
WRITE ON READ-ONLY FILE lfn AT addr.	Either the user attempted to write on a file with write interlock or the direct access file was not attached in WRITE mode.	Reattach file in write mode or clear write interlock.	CIO

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
WRITE OVER LABEL ILLEGAL ON lfn AT addr.	The user is not allowed to destroy the VOL1 label.	Use LISTLB control statement to obtain label data.	IMT
WRONG PARITY, lfn AT addr.	A 7-track tape is being read in opposite parity from which it was written.	Ensure accuracy of format parameter (F) on control statement or macro.	IMT
25555 FIELD LENGTH INCREASE.	The job field length was too small for LIBEDIT. Field length was increased to 26K.	None.	LIBEDIT
filenam ALREADY PERMANENT, AT addr.	The user has already saved or defined a file with the name specified.	Save or define file using different file name or purge existing file.	PFM
jobnam ASSIGNS EXCEED DEMANDS.	Dayfile message indicating that RESEX internal problems occurred. The resources actually assigned to the job jobnam exceed the resources demanded on a RESOURC statement.	Inform site analyst.	RESEX
n BAD FORMAT BLOCKS.	Block(s) not meeting X or SI-coded conversion requirements have been encountered on input.	Check tape format for errors.	TCOPY
pfn BUSY, AT addr.	The specified direct access file is attached in the opposite mode, or it is currently being accessed by one of the following. - More than 77B users in READ mode - More than 77B users in READAP mode - More than 7777B users in READMD mode	Reissue ATTACH until file becomes available, or issue ATTACH specifying NA option.	PFM
n DIRECTIVE ERRORS.	LIBEDIT could not interpret n directives because the records specified in the directives could not be found.	Correct errors as listed in LIBEDIT output and rerun job.	LIBEDIT
lfn EMPTY, AT addr.	The file specified on a SAVE request contains no data.	Verify that file contains data and retry.	PFM
nnn FILE(S) PROCESSED.	The operation was performed on nnn files.	None.	MFILES
xx FILE(S) RELEASED.	An xx number (octal count) of PRINT and/or PUNCH files were found and released from the user's control point.	None.	OUT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
jobnam MISSING RESOURCE.	Dayfile message indicating that RESEX internal problem occurred. RESEX expected but did not find a resource unit assigned to the specified job. This could occur if MAGNET was stopped while tapes were assigned.	Inform site analyst.	RESEX
n NOISE BLOCKS DELETED. n NOISE BLOCKS PADDED.	Block(s) on S or L output tape smaller than noise size have been deleted/padded.	None.	COPY
filenam NOT DECLARED RANDOM.	An EOF was encountered on the nonrandom file, filenam.	Verify that file name is in correct format.	LIBEDIT
filenam NOT FOUND.	RESTART was unable to retrieve a file named, but not included, on filenam.	Verify that filenam is valid.	RESTART
proc NOT FOUND.	Procedure specified : CALL statement cannot be found.	Verify that procedure name is correct and retry.	CONTROL
pfn NOT FOUND, AT addr.	One of the following. - The specified permanent file could not be found. - The specified user number could not be found. - The user is not allowed to access the specified file. - The user issued an indirect access file command on a direct access file. - The user issued a direct access file command on an indirect access file. If this message occurs in response to the SAVE request, the specified local file is not attached to the control point, is a direct access file, or is an execute-only file.	Verify that file name/ user number is correct, that access permission has been granted, and that correct access is being attempted.	PFM
xxx NOT IN PP LIB.	Dayfile message indicating that PP package xxx was not found in PP libraries.	Ensure that the correct PP package name was specified.	SFP
xxx NOT IN PP LIB. CALLED BY yyy.	Dayfile message indicating that PP package xxx, which was called by package yyy, was not found in the PP libraries.	Ensure that the correct PP package name was specified or inform site analyst.	SFP
filenam NOT ON MASS STORAGE, AT addr.	The file to be saved is not on mass storage; the first track of the file is not recognizable.	Verify that file is on mass storage.	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
n PARITY/BLOCK TOO LARGE ERRORS.	Parity and/or block-too-large errors have been encountered on the input file during the copy operation.	If dayfile shows block-too-large errors have occurred and tape is S, L, or F format, increase block size and retry; otherwise, tape is probably assigned in the wrong format. If parity errors have occurred, the tape is bad and the data on it cannot be correctly recovered.	COPY, TCOPY
nnnnn RECORDS CONVERTED.	Informative message indicating the number of records (nnnnn) converted from one character set to another.	None.	CONVERT
n RECORDS NOT REPLACED.	Informative message. LIBEDIT encountered n records on a replacement file that were not named in the directives and did not replace old file records.	Either change the directives so that the replacement file is a no-replace file or include an *IGNORE directive listing the records that are not to be used.	LIBEDIT
n RECORDS SPLITS OCCURRED.	Multiple blocks per record have been written on an S or L output tape.	None.	COPY
jobnam RESTARTED FROM yy/mm/dd. hh.mm.ss.	The checkpoint job identified by jobnam was restarted from the checkpoint taken on the specified date and time. This message is issued whenever a checkpoint job is restarted.	None.	RESTART
jobnam SHARE TABLE MISMATCH.	Dayfile message indicating that RESEX internal problem occurred. While processing the specified job, an expected share table entry match with the environment did not occur.	Inform site analyst.	RESEX

GLOSSARY C

Alphanumeric	The letters of the alphabet (A-Z) and the numeric digits (0-9).
ASCII	American National Standard Code for Information Interchange. The standard character set code and set used for information interchange between systems. ASCII is the default code set for 9-track coded tapes.
Block	Data recorded between magnetic tape interrecord gaps; a tape PRU. Blocking groups user records for greater transfer efficiency.
BOI	Beginning-of-information; the point in a file before which no file data exists.
Byte	One of the five groups of 12 bits within a 60-bit central memory word.
Character	A symbol whose coded representation may be processed within the system. Refer to Alphanumeric, Graphic, and Control character.
Checkpoint file	File on which the results of a partially completed job are dumped when a CKP control statement is processed.
CIO	Combined Input/Output. System routine that performs NOS I/O.
Control character	A character whose occurrence within a print file causes an action. Examples are the carriage return and line feed characters.
Control statement	A sequence of words and characters that call a system routine to perform a job step. The control statement must conform to format specifications and end with either a period or a right parenthesis.
Control statement record	1. A record containing control statements, such as a procedure file. 2. The first record of a job containing the sequence of control statements that specifies all steps for job execution
CYBER Loader	The NOS product that prepares programs for execution by placing program instruction and data blocks in central memory.

CYBER Record Manager (CRM)	The NOS product that acts as an I/O interface between CIO and other NOS products including ALGOL 4, ALGOL 5, COBOL 5, DMS-170, FORTRAN Extended 4, FORTRAN 5, and PL/I. The CRM functions are split between two file processors, Basic Access Methods (BAM) and Advanced Access Methods (AAM). Refer to the CRM manuals listed in the preface for definitions of CRM file organizations, block types, and record types.
Direct access file	A permanent mass storage file that can be assigned to a user's job. All changes to this file are made on the file itself rather than a working copy of the file (refer to Indirect access file).
Display code	Six-bit code set used to represent characters in central memory and in binary files. Appendix A translates the code set.
EBCDIC	Extended Binary Coded Decimal Interchange Code. An 8-bit code set that, if the user requests, NOS uses to read or record data on 9-track tapes.
Empty PRU/record	Refer to Zero-length PRU.
Entry point	A location within a program that can be referenced by name from other programs.
EOF	End-of-file indicator; the file may be part of a multifile file whose end is defined by an EOI indicator.
EOI	End-of-information indicator; marks the end of a named file.
EOR	End-of-record indicator; marks the end of a logical record.
EOT	End-of-tape; metallic strip marking the end of the recordable portion of a magnetic tape.
FET	File Environment Table; a table used by a COMPASS programmer to define and interrogate the current status and properties of a file assigned to a job.
File	1. Data that begins at BOI and ends at an EOI and is referenced by a 1- to 7-alphanumeric character name. 2. A portion of a multifile file ending with an EOF. 3. Data recorded on a magnetic tape beginning after an HDR1 label and ending before an EOF1 label. NOS control statements requiring a file name lfn (except tape assignment statements) refer to definition 1; NOS control statements that have a parameter specifying the number of files refer to definition 2. Definition 3 applies only to labeled magnetic tapes.

File set

One or more tape files referred to by the lfn on a tape assignment statement. A file set may consist of:

- One file recorded on a single volume.
- More than one file recorded on a single volume.
- One file recorded on more than one volume.
- More than one file recorded on more than one volume.

To conform to the ANSI tape standard, all files within a file set must have the same setid in their HDR1 labels.

File type

A category of files handled similarly by the system. Section 2 categorizes NOS files according to permanent file types and types of files assigned to user jobs.

FIT

File Information Table. The table that defines a file for access by CYBER Record Manager.

FNT/FST entry

File Name Table/File Status Table. Two-word system table description of a file currently assigned to a job.

Frame

A tape recording unit made up of one bit from each tape track (7 bits for 7-track tape and 9 bits for 9-track tape). Each frame on a coded tape usually represents 1 character.

Generation

The position of a file within a series of files, each file developed from the preceding file. The generation number and generation version number of a tape file can be entered in its HDR1 label.

Graphic

A character that can be printed or displayed. Refer to Control character.

Indirect access file

A mass storage permanent file. Indirect access files can be accessed only through a working copy of the file. If requested, the working copy replaces the permanent file.

Input file type

Job file. Its first record is a control statement record which may be followed by records containing data, directives, or programs used by job steps.

Interrecord gap

Space skipped between the writing of data blocks on magnetic tape.

Job

A set of control statements and the data and directives used by those statements. A batch job must begin with the job and USER statements. A time-sharing job begins at the user's login to a terminal.

Job step	One of the sequence of operations performed within a job. Usually, each control statement results in one job step. However, a load sequence is a single job step that may result from several loader control statements.
Label	An 80-character block written to identify and/or delimit a tape volume or a file.
Level number	Octal number within a PRU terminator indicating the type of boundary it represents. A level 17 in an empty PRU is a NOS EOF; a level 0 in a short PRU is a NOS EOR; a level 1 in a short PRU is an EOR from an interactive terminal; and a level 16 in a short PRU is an EOF on a checkpoint file.
lfn	Name of a file assigned to the job.
Library	1. A collection of programs or routines. 2. A file containing records that are accessed individually. 3. A file searched by CYBER Loader for entry points referenced by a program. 4. A file containing compressed records in Modify or Update format.
Library file	A read-only file that can be accessed by several users simultaneously.
Line	A unit data terminated by a zero-byte terminator. Unit used time-sharing I/O, line printer output, and card reader inp'
Load point	Metallic strip marking the beginning of the recordable portion of a magnetic tape.
Local file	1. A file type that refers to a temporary file other than the primary file. It often contains a copy of an indirect access file or data from a magnetic tape. 2. A file currently assigned to a job.
Logical record	A unit of data ending with an EOR.
Mass storage	Magnetic disk or extended memory.
Multifile file	A file containing more than one logical file. It begins at BOI and ends at EOI. On a labeled tape, a multifile file is delimited by corresponding HDR1 and EOF1 labels.
Multifile set	A tape file set having more than one tape file.
Noise	Any tape block less than the minimum acceptable block size for its data format. Noise is discarded by the system.
Nonallocatable devices	Magnetic tape unit, card reader, card punch, or line printer.

Parity	Error detection method for which an extra bit is written per byte (or tape frame) so that each byte always has either an odd or an even number of bits set. Parity errors are detected when the recorded data is read.
Primary file	The temporary file designated as the default file by either the PRIMARY, NEW, or OLD control statements.
Print file	Output file containing data to be printed. Printing occurs at job completion or upon execution of an OUT, ROUTE, or DISPOSE statement.
PRU	Physical record unit. Data unit transmitted by a single read or write to a device.
Punch file	Output file containing data to be punched on cards.
Random access file	A file whose records are accessed through a directory containing the address of each record.
Record	An individually accessible unit of information. Its further definition depends on its context. Physical records, logical records, and CRM records can have different delimiters.
Rollout file	File containing all information concerning a job whose execution has been temporarily suspended.
Sequence number	1. Number at the beginning of each line of a file. 2. For tape labels, number indicating position of a file within a file set.
Sequential access file	File whose records are accessed in the order in which they appear.
Short PRU	A physical record unit containing less than the maximum data for a unit.
Tape mark	Bit sequence written by a tape unit indicating a tape boundary.
Timed/event rollout file	A rollout file whose job is returned to execution when a requested event has occurred or a specified time period has elapsed.
Volume	A reel of magnetic tape.
VSN	Volume serial number. A 1- to 6-character sequence uniquely identifying a reel of tape.
Working file	A temporary file currently assigned to a job.
Write ring	A circular device inserted into a tape reel indicating to the tape unit that it can write on that reel. NOS checks for the presence of a write ring if the user requested it.

Zero byte terminator	Twelve zero bits in the lowest byte of a central memory word that terminate a line. Because the display code for a colon is 6 bits of zero, double colons should be avoided. If stored in the lowest byte of a central memory word, the colons are treated as a zero-byte terminator. Files used with the COPYCF, COPYCR, COPYSBF, LIST80, L072, RESEQ, ROUTE, and SUBMIT control statements must contain zero-byte terminated lines. Terminal input consists of zero-byte terminated lines.
Zero-length PRU/record	A PRU terminator written immediately after another PRU terminator; a PRU that contains no user data.
6/7/8/9 multipunch	Signifies a card deck EOI.
6/7/9 multipunch	Signifies a card deck EOF.
7/8/9 multipunch	Signifies a card deck EOR.

SAMPLE JOB OUTPUT D

Appendix D lists the output information printed for the sample job shown below. The notes in the right margin identify the various format conventions of NOS output. The job consists of the following statements.

```
TESTA.
USER(JEANCOM,PASSWRD)
CHARGE(JLC3951,27)
FTN.
/EOR
      PROGRAM CONVER(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C
C        INPUT -- 10-DIGIT OCTAL NUMBER, SUCH AS 0000000177
C        OUTPUT -- DECIMAL EQUIVALENT OF OCTAL NUMBER INPUT
C        IF AN 8 OR 9 DIGIT IS ENTERED, THE PROGRAM STOPS.
C        IF A NON-NUMERIC CHARACTER IS ENTERED, THE PROGRAM ABORTS.
C
      INTEGER NUM,DIGIT,SUM,PLVAL,PLACE
      DIMENSION NUM(10)
C
C     READS AN OCTAL NUMBER INTO THE NUM ARRAY
C
   10 READ(5,1000) NUM
 1000 FORMAT(10I1)
      IF (EOF(5)) 50,15
C
C     CHECKS IF AN 8 OR 9 WAS INPUT
C
   15 DO 20 I=1,10
   20 IF (NUM(I).EQ.8 .OR. NUM(I).EQ.9)   GO TO 50
C
C     CONVERTS THE OCTAL NUMBER BY MULTIPLYING EACH DIGIT
C     BY ITS PLACE VALUE AND ADDING IT TO A SUM
C
      DIGIT=10
      PLACE=0
      SUM=NUM(DIGIT)
   30 DIGIT=DIGIT-1
      PLACE=PLACE+1
      IF (DIGIT.EQ.0)    GO TO 40
      PLVAL=NUM(DIGIT) * 8 ** PLACE
      SUM=SUM+PLVAL
      GO TO 30
C
C     OUTPUTS CONVERTED NUMBER
C
   40 WRITE(6,4000) SUM
 4000 FORMAT(1X,1I10)
      GO TO 10
   50 STOP
      END
/EOR
/EOF
```

```
NOS 1                      yy/mm/dd.
          OPERATING SYSTEM
        JOB ORIGIN = BATCH.
```

The first three lines of the banner page indicate that this local batch job was run under the control of the Network Operating System. The system creation date is specified by yy/mm/dd. (year/month/day.).

```
      USER NUMBER = JEANCOM
     JOBCARD NAME = TESTA00
```

The user number is that which was supplied on the USER statement. The jobcard name is the name of the particular job which was supplied on the job statement.

```
 AAAAAAAAAA    AAAAAAAAAA   FFFFFFFFFFFF  IIIIIIIIIIII   AAAAAAAAAA    JJJJJJJJJJJJ   CCCCCCCCCC
AAAAAAAAAAAA  AAAAAAAAAAAA  FFFFFFFFFFFF  IIIIIIIIIIII  AAAAAAAAAAAA   JJJJJJJJJJJJ  CCCCCCCCCCCC
AA        AA  AA        AA  FF                 II       AA        AA        JJ       CC        CC
AA        AA  AA        AA  FF                 II       AA        AA        JJ       CC
AA        AA  AA        AA  FF                 II       AA        AA        JJ       CC
AA        AA  AA        AA  FF                 II       AA        AA        JJ       CC
AA        AA  AA        AA  FF                 II       AA        AA        JJ       CC
AA        AA  AA        AA  FFFFFFFF           II       AA        AA        JJ       CC
AAAAAAAAAAAA  AAAAAAAAAAAA  FFFFFFFF           II       AAAAAAAAAAAA        JJ       CC
AAAAAAAAAAAA  AAAAAAAAAAAA  FF                 II       AAAAAAAAAAAA        JJ       CC
AA        AA  AA        AA  FF                 II       AA        AA        JJ       CC
AA        AA  AA        AA  FF                 II       AA        AA        JJ       CC
AA        AA  AA        AA  FF                 II       AA        AA        JJ       CC
AA        AA  AA        AA  FF                 II       AA        AA   JJ   JJ       CC        CC
AA        AA  AA        AA  FF            IIIIIIIIIIII  AA        AA   JJJJJJ        CCCCCCCCCCCC
AA        AA  AA        AA  FF            IIIIIIIIIIII  AA        AA    JJJJ          CCCCCCCCCC
```

The first 4 characters of the banner job name are generated from the user index associated with the user number. These 4 characters are unique to each user and remain the same for subsequent jobs run under the same user number. The last 3 characters are the job sequence number assigned by the system at the time of processing.

```
yy/mm/dd.  hh.mm.ss.
```

This line specifies the current date (year/month/day.) and the time (hours. minutes. seconds.) when job printing was initiated.

Processing of the FTN. control statement compiles the source program in the second record of the job. The following is the program listing and load map produced by that compilation.

```
      PROGRAM CONVER     73/74   OPT=1                    FTN 4.8+497       79/05/14. 09.55.13       PAGE     1

 1              PROGRAM CONVER(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
               C
               C       INPUT -- 10-DIGIT OCTAL NUMBER, SUCH AS 0000000177
               C       OUTPUT -- DECIMAL EQUIVALENT OF OCTAL NUMBER INPUT
 5             C       IF AN 8 OR 9 DIGIT IS ENTERED, THE PROGRAM STOPS.
               C       IF A NON-NUMERIC CHARACTER IS ENTERED, THE PROGRAM ABORTS.
               C
                     INTEGER NUM,DIGIT,SUM,PLVAL,PLACE
                     DIMENSION NUM(10)
10             C
               C       READS AN OCTAL NUMBER INTO THE NUM ARRAY
               C
                  10 READ(5,1000) NUM
                1000 FORMAT(10I1)
15                   IF (EOF(5)) 50,15
               C
               C       CHECKS IF AN 8 OR 9 WAS INPUT
               C
                  15 DO 20 I=1,10
20                20 IF (NUM(I).EQ.8 .OR. NUM(I).EQ.9)  GO TO 50
               C
               C       CONVERTS THE OCTAL NUMBER BY MULTIPLYING EACH DIGIT
               C       BY ITS PLACE VALUE AND ADDING IT TO A SUM
               C
25                   DIGIT=10
                     PLACE=0
                     SUM=NUM(DIGIT)
                  30 DIGIT=DIGIT-1
                     PLACE=PLACE+1
30                   IF (DIGIT.EQ.0)  GO TO 40
                     PLVAL=NUM(DIGIT) * 8 ** PLACE
                     SUM=SUM+PLVAL
                     GO TO 30
               C
35             C       OUTPUTS CONVERTED NUMBER
               C
                  40 WRITE(6,4000) SUM
                4000 FORMAT(1X,I10)
                     GO TO 10
40                50 STOP
                     END


      SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
 4141  CONVER

VARIABLES     SN  TYPE          RELOCATION
 4220  DIGIT     INTEGER                        4224  I         INTEGER
 4225  NUM       INTEGER    ARRAY               4223  PLACE     INTEGER
 4222  PLVAL     INTEGER                        4221  SUM       INTEGER
FILE NAMES       MODE
    0  INPUT                2054  OUTPUT                0  TAPE5     FMT          2054  TAPE6     FMT

EXTERNALS         TYPE   ARGS
      EOF        REAL      1

STATEMENT LABELS
 4142  10                              0  15          INACTIVE              0  20
 4162  30                           4173  40                           4176  50
 4206  1000    FMT                  4214  4000    FMT

LOOPS  LABEL    INDEX     FROM-TO    LENGTH    PROPERTIES
 4150  20       I           19 20       7B     INSTACK    EXITS

STATISTICS
  PROGRAM LENGTH              353B     235
  BUFFER LENGTH              3664B    1972
        52000B CM USED
```

```
09.55.12.TESTA.
09.55.12.USER(JEANCOM,)
09.55.13.CHARGE(JLC3951,27)
09.55.13.FTN.
09.55.14.       .075 CP SECONDS COMPILATION TIME
09.55.14.UEAD,       0.002KUNS.
09.55.14.UEPF,       0.005KUNS.
09.55.14.UEMS,       0.703KUNS.
09.55.14.UECP,       0.082SECS.
09.55.14.AESR,       2.348UNTS.
09.55.21.UCLP, 6232,      0.192KLNS.
```

This line specifies the job name, the current date, and the computer system being used. The dayfile includes a listing of the control statements, system-supplied status messages, and program output, if any. Spaces precede status messages and program output. Each line includes the time the message was issued to the dayfile.

The last six lines specify the type and amount of system resources the job used. This job used 0.002 kilounit of application activity, 0.005 kilounit of permanent file activity, 0.703 kilounit of mass storage activity, 0.082 second of central processor time, and 2.348 system resource units. The 6232 after UCLP gives the machine ID as 62 and the EST ordinal of the printer as 32. The job produced 0.192 kiloline (192 lines) of printable output.

Depending on the resources used, additional information may be included in the dayfile. Refer to Job Completion in section 3 for the formats of these messages.

TIME-SHARING INTERFACE E

The time-sharing interface, IAF or the time-sharing executive, processes communications between NOS and time-sharing terminals. When in the batch subsystem, the time-sharing user can enter any of the control statements described in this manual. He can also enter a number of control statements and commands intended for use only by time-sharing jobs. The time-sharing control statements and commands are described in the IAF Reference Manual and the Time-Sharing User's Reference Manual.

Tables 1-A-1 and 1-A-2 list the time-sharing character sets. This appendix describes terminal character code conversion and the time-sharing control statements that can be included in procedure files.

TERMINAL CHARACTER CONVERSION

Normal input mode from an ASCII code terminal uses a 63- or 64-character set where all lowercase alphabetics are converted to uppercase characters. Under ASCII mode, the characters 74 and 76 represent the beginning of a 74xx or 76xx escape sequence. Under normal mode, the characters 74 and 76 are treated as data rather than escape codes. ASCII and normal modes apply to both input and output.

DATA INPUT

The manner in which the characters entered from a terminal are interpreted by the system depends on whether the user requests ASCII or NORMAL mode. For example, if the user enters the following characters when in ASCII mode.

aAbBcCdDeEfF

The central memory equivalent is:†

59		47		35		23		11	0
76	01	01	76	02	02	76	03	03	76
04	04	76	05	05	76	06	06	00	00

However, if in NORMAL mode when the characters are entered, the characters are mapped into the 64-character subset of the ASCII character set that contains only uppercase letters; then the central memory equivalent is:

59		47		35		23		11	0
01	01	02	02	03	03	04	04	05	05
06	06	00	00	00	00	00	00	00	00

Refer to appendix A for further description of the time-sharing code sets.

†Partial words are zero-filled; partial bytes are blank-filled.

DATA OUTPUT

Data output is in either a 64/63- or 128-character set, depending on whether the terminal is in normal or ASCII mode. When the terminal is in normal mode, the codes 74 and 76 represent data rather than escape codes. In ASCII mode, 74 and 76 are treated as the beginning of an escape sequence.

For a more detailed description of terminal operation, refer to the IAF Reference Manual or the Time-Sharing User's Reference Manual.

Data can also be transmitted to or from a terminal through a paper tape reader.

TIME-SHARING CONTROL STATEMENTS

The user can enter the following control statements in a time-sharing job. For more information on these statements, refer to the IAF Reference Manual or the Time-Sharing User's Reference Manual.

NOTE

If a time-sharing control statement is included in a non-time-sharing job, the system terminates the job.

ASCII STATEMENT

The ASCII control statement specifies that further terminal input and output is to be interpreted as 6/12 display code.

The control statement format is:

```
ASCII.
```

If this control statement is processed while output is still available, the terminal switches to ASCII mode for the remainder of the output.

CSET STATEMENT

The CSET control statement specifies the current code set of the terminal.

The control statement format is:

```
CSET(m)
```

m	Current terminal code set; m may be one of the following.	
	ASCII	Uses 6/12 display code set; escape code processing.
	NORMAL	Uses display code set; escape code processing is disabled.

If this control statement is processed while output is still available, the terminal switches to the new character set mode for the remainder of the output.

PARITY STATEMENT†

The PARITY control statement sets the terminal to the indicated parity.

The control statement format is:

PARITY(p)

p — Terminal parity; p may be one of the following.

ODD	Set odd parity.
EVEN	Set even parity.

If p is omitted, odd parity is assumed.

If this control statement is processed while output is still available, the terminal parity switches to the new parity for the remainder of the output.

TRMDEF STATEMENT††

When executed by a time-sharing job, the TRMDEF control statement specifies changes in the characteristics of the network terminal that issued the statement. For a detailed description of this statement, refer to the IAF Reference Manual.

The control statement format is:

TRMDEF(L=lfn, xx_1=$value_1$, xx_2=$value_2$, . . . , xx_n=$value_n$)

L=lfn — Names optional local file to receive the terminal redefinition information. If it names a file other than the output file for the terminal, the terminal characteristics do not change until the file lfn is copied to the output file for the terminal.

xx_1=$value_i$ — A 2-character parameter mnemonic equated to a legal value for that terminal characteristic. The legal value may be entered in one of the following formats.

y	Alphanumeric character.
y	Any character, including special characters.
yyyB	Octal value of an ASCII character.
Xyy	Hexadecimal value of an ASCII character.

The parameter mnemonics and the legal values for terminal characteristics are listed in the IAF Reference Manual.

†Not available for IAF.
††Valid only from IAF.

CARD FILE DATA CONVERSION F

Data within the system is stored in binary or coded format. Binary data is variable length central memory images. Coded data consists of display-coded characters. Each coded line is stored as an even number of characters. If an odd number of characters is entered, the system appends a space to make it even.

This appendix describes the formats for punch cards. It also describes the conversion performed by the system on data transferred between the system and card readers and punches.

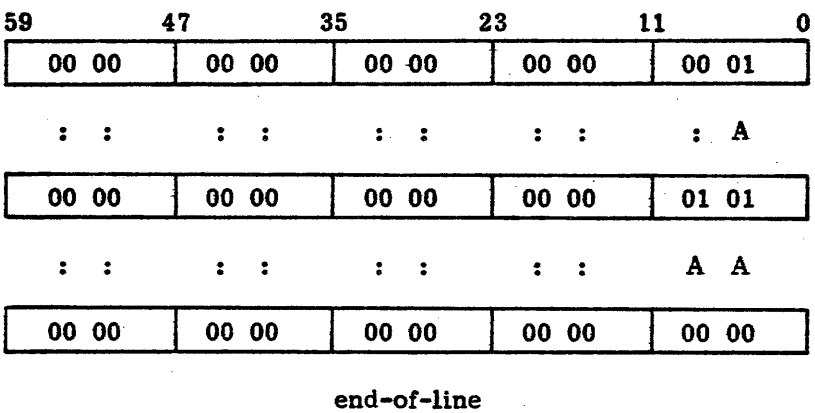
When using the 64-character set, the user should avoid using consecutive colons (00 characters). It is possible for these colons to be interpreted as an end-of-line. An end-of-line is defined as 12 to 66 zero bits, right-justified in one or two central memory words. If consecutive colons appear in the lower 12 bits of a central memory word, they are interpreted as an end-of-line rather than as colons.

Example:

The following characters are punched on a coded card beginning in column 1.

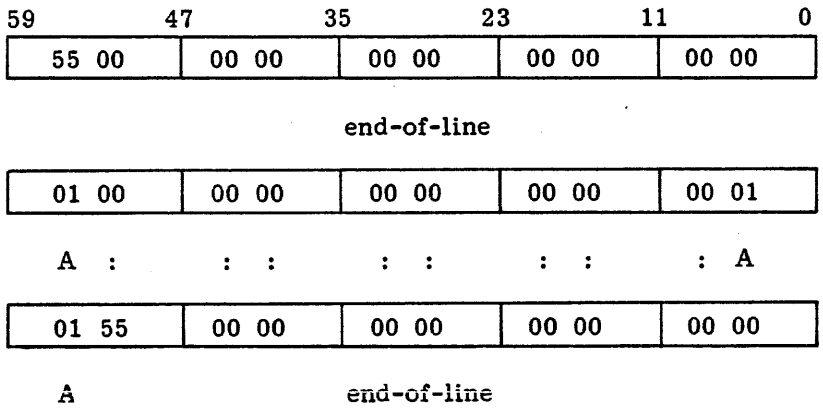
::::::::A::::::::AA

This appears in memory as follows:



end-of-line

However, if the characters were copied with the COPYSBF utility, the following appears.



Because the COPYSBF utility shifts each line one 6-bit character to the right and adds a space, copying nine colons puts 12 zero bits in the last byte of the first word. This is interpreted as an end-of-line.

NOTE

If a colon is the last character of an input line, the system appends a space to perserve the colon and then appends an end-of-line. If needed, a second space is added to ensure an even number of 6-bit characters. Refer to figure 1-F-1.

INPUT CARD FILE FORMATS

The system reads cards in coded and binary formats. The following conditions apply in both formats.

- A card with a 7/8/9 punched in column 1 is an EOR mark.
- A card with a 6/7/9 punched in column 1 is an EOF† mark.
- A card with a 6/7/8/9 punched in column 1 is an EOI mark.

The remainder of each card is ignored except for columns 79 and 80 of the EOR and EOF cards. These columns can contain the keypunch conversion mode for the input records that follow.†† Conversion modes are discussed in Coded Cards.

CODED CARDS

Cards are read in Hollerith punch code. The 3447 card reader controller converts the Hollerith code to internal BCD code and passes the data to the card reader driver. The driver converts the data from internal BCD code to display code. Up to 80 characters can be transferred per card. Trailing blank bytes are deleted. If a line has

†The 6/7/9 keypunch mark is not supported by either Export/Import or RBF.
††HASP terminals can support other forms of separator cards such as /*EOR and /*EOI. (Refer to the RBF Reference Manual.)

an odd number of characters, one trailing blank is added to make it even. In order to preserve the colon (00_8) of the 64-character set, a trailing blank byte is either retained or appended as the last character in an even line. Examples of coded card conversion are shown in figure 1-F-1.

Conversion Modes

Two conversion modes, O26 and O29,† exist for the Hollerith punch code. All data is converted in the system default keypunch mode unless a conversion mode change is specified. This change can be specified on any of the following cards.

The job card, 7/8/9 card (EOR mark), and 6/7/9 (EOF mark) can contain the keypunch conversion mode in columns 79 and 80. A 26 punched in columns 79 and 80 indicates that all subsequent coded cards are converted in O26 mode. A 29 indicates that subsequent cards are converted to O29 mode. Each conversion change remains in effect until another change card is encountered or the job ends. The user can switch between O26 and O29 mode as often as desired. If 26 or 29 does not appear in columns 79 and 80 of the job card, the initial keypunch mode of that job is the system default mode. If 26 or 29 does not appear on a 7/8/9 or 6/7/9 card, no conversion change is made, and the most recent keypunch mode remains in effect.

Keypunch mode can also be changed by a card containing a 5/7/9 punch in column 1. A blank (no punch) in column 2 indicates O26 conversion mode; a 9 punch in column 2 indicates O29 mode. The conversion change remains in effect until another change card is encountered or the job ends.

The 5/7/9 card also allows literal input when 4/5/6/7/8/9 is punched in column 2. Literal input allows 80-column binary data to be read while transmitting input in coded mode. Cards are read (16 central memory words per card) until a card identical to the previous 5/7/9 card (4/5/6/7/8/9 in column 2) is read. The next card can then specify the new conversion mode.

In order to maintain system integrity, an end-of-information card always terminates 80-column binary input (lieteral input). Either of the following is interpreted as an end-of-information card even though it appears in a literal input record.

- A card with 6/7/8/9 punched in column 1 and with columns 2 through 80 blank.
- A card with 6/7/8/9 punched in columns 1 and 80 and with columns 2 through 39 and columns 41 through 79 blank.

† These codes are ignored by a 200 User Terminal since conversion mode is selected by a hardware switch. (Refer to the Export/Import Reference Manual and the Remote Batch Facility Reference Manual.)

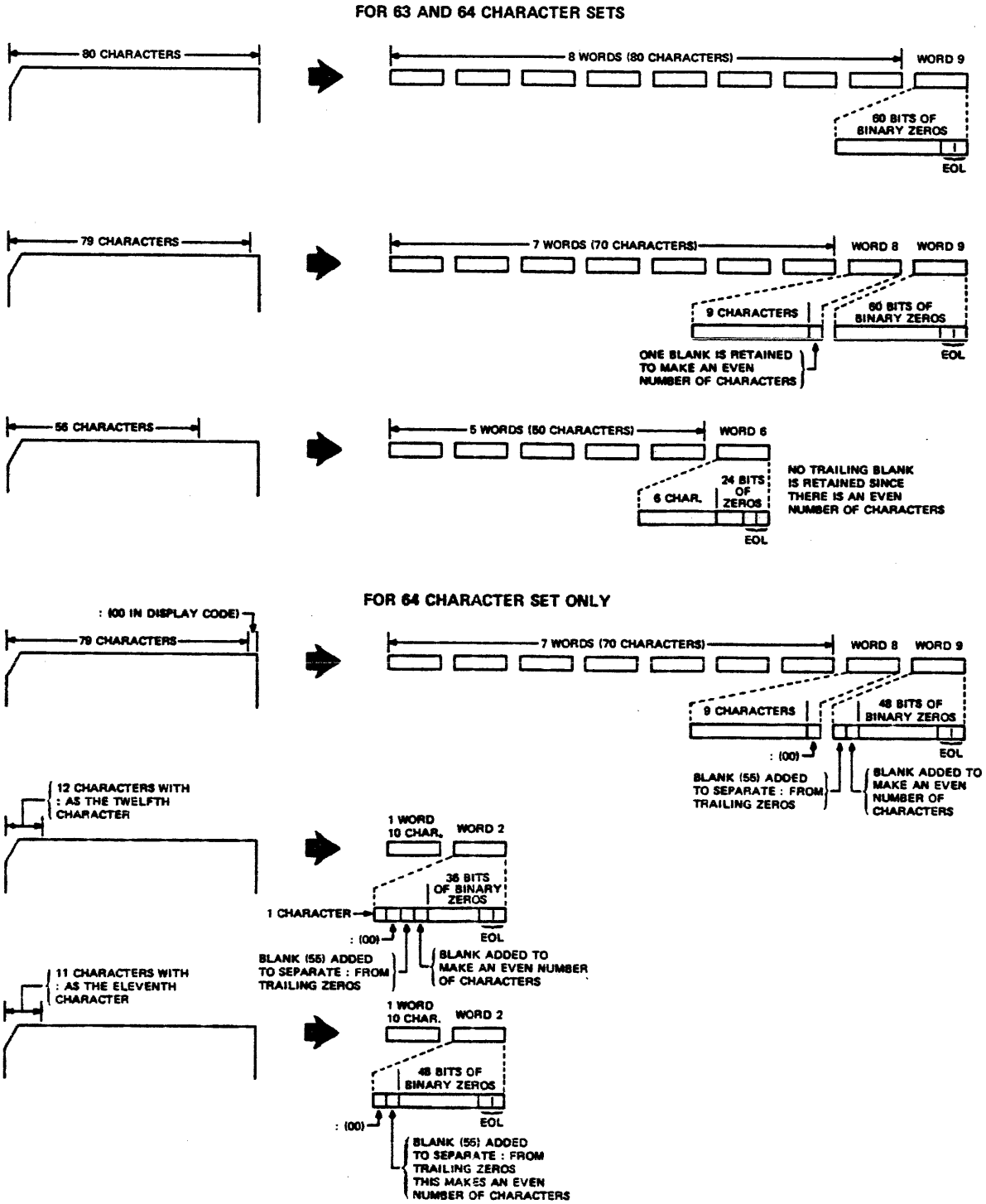


Figure 1-F-1. Examples of Coded Card Conversion

BINARY CARDS

Binary cards are denoted by a 7/9 punch in column 1 and can contain up to 15 central memory words. The 3447 card reader controller reads the binary data and passes it to the card reader driver in 12-bit codes. Each card column row corresponds to a bit position. The driver checks the checksum figure if this option is specified. The driver then passes the data to the central memory buffer.

The fields within a binary card are:

Column(s)	Description
1	7/9 punch indicates a binary card.
	4 punch ignores checksum punch in column 2.
	Rows 0, 1, 2, and 3 contain the binary equivalent of the word count of the card.
2	Binary data checksum (modulo 4095).
3 through 77	15 central memory words of binary data.
78	Blank.
79 and 80	24-bit binary card sequence number.

SUMMARY

The following punches appearing in column 1 of a card have the corresponding meaning to the card reader driver.

Punch	Represents
7/8/9	End-of-record (optional conversion mode change).
6/7/9	End-of-file (optional conversion mode change).
6/7/8/9	End-of-information.
5/7/9	Conversion mode change/read 80-column binary.
7/9	Binary card.
Not 7 and 9	Coded card.

PUNCH FILE FORMATS

Punched cards can be in three formats.

- Coded (Hollerith punch)
- Binary
- Absolute binary

The following conditions apply to all three formats.

- When an EOR is encountered, a card is punched with a 7/8/9 in columns 1 and 80. This card is offset.
- When an EOF is encountered for a file, a card is punched with a 6/7/9 in columns 1 and 80; the remainder of the card is blank. This card is offset.
- When an EOI is encountered on a file, a card is punched with a 6/7/8/9 in columns 1 and 80; the remainder of the card is blank. This card is offset.
- If a compare error is encountered, the erroneous card and the following card are offset. These two cards are repunched until no error is detected. An EOI card with 6/7/8/9 punches in columns 1 and 80 contains a binary count in column 40 of the number of compare errors.
- During the punching of each file, the system maintains a count of the number of cards punched for the file. If the number exceeds the limit for which the user is validated, punching of the file is terminated. A special banner card with the word LIMIT is punched and offset as the last card of the deck.

The following methods are used by the system to punch each of the three forms of cards.

CODED CARDS (PUNCH)

With the exception of decks punched via the DISPOSE request, the keypunch mode (O26 or O29) of coded cards depends on the job origin type. If the job is of local batch origin, decks are punched in the initial keypunch mode (that is, the mode specified on the job card or set by system default). For all other job origin types, decks are punched in the system default keypunch mode. However, the DISPOSE request allows the user to specify that decks be punched in either O26 or O29 mode, regardless of the job's keypunch mode.

BINARY CARDS (PUNCHB)

The card punch driver retrieves 15 words of binary data from central memory. The driver then generates a checksum for the data and issues a card number. The card punch controller receives the binary data and punches it on the card unchanged, that is, in 12-bit codes. Each row in a card column corresponds to a bit position. The driver formats the binary card in the following manner.

Column(s)	Contents
1	7/9 punch denotes binary card. Rows 0, 1, 2, and 3 contain the binary equivalent of the word count of the card.
2	Binary data checksum (modulo 4095).
3 through 77	15 central memory words of binary data.
78	Blank.
79 and 80	24-bit binary card sequence number.

ABSOLUTE BINARY CARDS (P8)

Absolute binary cards are central memory images in 12-bit codes. Each row in a card column corresponds to a bit position. Sixteen central memory words are punched per card with no special punches or fields added.

ANSI TAPE LABEL FORMATS G

ANSI labels perform two functions. They provide information that uniquely identifies a file and the reel on which it resides, and they mark the BOI and EOI of a file and the beginning and end of a reel.

ANSI labels are designed to conform to the American National Standard Magnetic Tape Labels for Information Interchange X3.27-1969. All labels are 80 characters long and are recorded at the same density as the data on the tape. The first 3 characters of an ANSI label identify the label type. The fourth character indicates a number within a label type.

The following is a summary of each label type, name, function, and whether or not it is required.

Type	No.	Name	Used As	Required/Optional
VOL	1	Volume header label	Beginning-of-volume	Required
UVL	1-9	User volume label	Beginning-of-volume	Optional
HDR	1	File header label	Beginning-of-information	Required
HDR	2-9	File header label	Beginning-of-information	Optional
UHL	†	User header label	Beginning-of-information	Optional
EOF	1	End-of-file label	End-of-information	Required
EOF	2-9	End-of-file label	End-of-information	Optional
UTL	†	User trailer label	End-of-information	Optional
EOV	1	End-of-volume label	End-of-volume	Required when appropriate
EOV	2-9	End-of-volume label	End-of-volume	Optional

REQUIRED LABELS

The VOL1, HDR1, and EOF1 labels are required on all ANSI-labeled tapes. In addition, an EOV1 label is required if the physical end-of-tape reflector is encountered before an EOF1 label is written or if a multifile set is continued on another volume. In the descriptions of the contents of these labels, n is any numeric digit and a is any letter, digit, or any of the following special characters.

†Any member of the CDC 6-bit subset of the ASCII character set.

Δ	)	<
!	*	=
"	+	>
≠	,	?
$	-	@
%	.	[
&	'	\
/	:	]
(	;	^

Some fields are optional. An optional field which does not contain the designated information must contain blanks. Fields which are not described as optional are required and written as specified. n-type fields are right-justified and zero-filled, and a-type fields are left-justified and blank-filled.

VOL1 — VOLUME HEADER LABEL

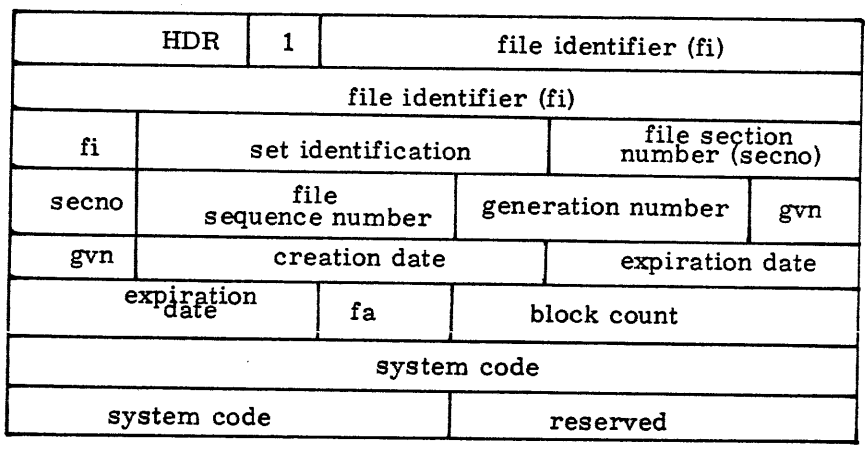
The volume header label must be the first label on a labeled tape. All reels begin with a VOL1 label. If two or more reels belong to a volume set, the file section field in the following HDR1 label gives the actual reel number.

VOL		1	volume serial number
va	reserved		
reserved			
reserved			owner identification
owner identification (oid)			
oid	reserved		
reserved			
reserved			lsl

Character Position	Field Name	Length (in characters)	Contents	Default	Checked on Read
1-3	Label identifier	3	Must be VOL.		Yes
4	Label number	1	Must be 1.		Yes
5-10	Volume serial number	6	Volume identification assigned by owner to identify this physical reel of tape	As read from existing label	Yes, if the file was assigned by volume serial number.
11	Accessibility (va)	1	An a character which indicates the restrictions, if any, on who may have access to the information on the tape. A blank means unlimited access. Any other character means special handling, in the manner agreed between the interchange parties. Refer to the BLANK control statement.	Blank (unlimited access)	No (refer to BLANK control statement).
12-31	Reserved for future standardization	20	Must be blanks.		No
32-37	Reserved for future standardization	6	Must be blanks.		No
38-51	Owner identification (oid)	14	Any a characters identifying the owner of the physical volume.	family name, user number	Refer to discussion of fa field of HDR1.
52-79	Reserved for future standardization	28	Must be blanks.		No
80	Label standard level (lsl)	1	1 means the labels and data formats on this volume conform to the requirements of the ANSI standard. A blank means the labels and data formats on this volume require the agreement of the interchange parties.	1	No

HDR1 — FIRST FILE HEADER LABEL

The first file header label must appear before each file. When a file is continued on more than one volume, the file header label is repeated after the volume header label on each new volume for that file. If two or more files are grouped in a multifile set, each HDR1 label indicates the relative position of its associated file within the set.



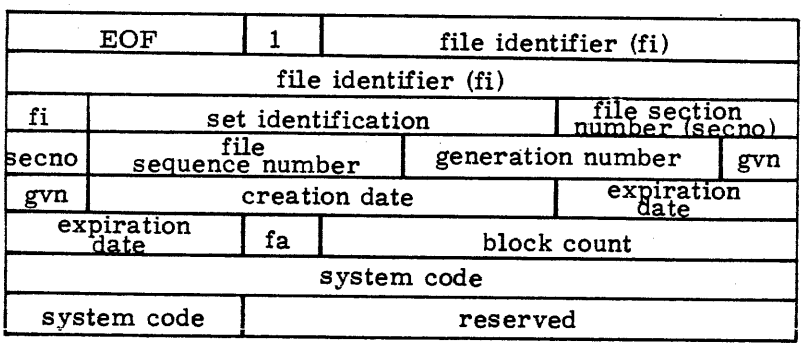
Character Position	Field Name	Length (in characters)	Contents	Default	Checked on Read
1-3	Label identifier	3	Must be HDR.		Yes
4	Label number	1	Must be 1.		Yes
5-21	File identifier (fi)	17	Up to 17 a characters used as the file identification (fileid) parameter on the LABEL control statement.	Blank	Checked if specified.
22-27	Set identification	6	Up to 6 a characters used as the setid parameter on the LABEL control statement. To conform to the ANSI tape standard, this value should be the same for all files of a multifile set.	Blank	Checked if specified.
28-31	File section number (secno)	4	Four n characters identifying the file section number. The file section number of the first HDR1 label of a file is 0001. If the file extends to more than one volume, this number is incremented by one for each subsequent volume. This value corresponds to the secno parameter on the LABEL statement.	0001	Checked if specified.
32-35	File sequence number	4	Four n characters used as the seqno parameter on the LABEL statement. This parameter specifies the position of a file within a file set. This value is 0001 for the first file, 0002 for the second, and so on. In all the labels for a given file, this field contains the same number.	0001	Checked if specified.

Character Position	Field Name	Length (in characters)	Contents	Default	Checked on Read
36-39	Generation number (optional)	4	Four n characters specifying the generation number of a file. This is the genno parameter of the LABEL statement. This value is 0001 for the first generation of a file, 0002 for the second, and so on.	0001	Checked if specified.
40-41	Generation version number (gvn)	2	Two n characters used to distinguish successive iterations of the same generation. The generation version number of the first attempt to create a file is 00. This value corresponds to the gvn parameter of the LABEL control statement.	00	Yes
42-47	Creation date	6	Date the file was created; it is recorded as a space followed by two n characters for the year followed by three n characters for the day within the year. This value corresponds to the cdate parameter of the LABEL control statement.	Current date	Yes. The creation date is meaningful only on read operations; on write operations, the current date is always used.
48-53	Expiration date	6	The file is considered expired when today's date is equal to or later than the date given in this field. When this condition is satisfied, the remainder of the volume may be overwritten. Thus, to be effective on multifile volumes, the expiration date of a file must be less than or equal to the expiration date of all preceding files on the volume. The expiration date is written in the same format as the creation date.	Current date	Checked if write attempted.

Character Position	Field Name	Length (in characters)	Contents	Default	Checked on Read
			It corresponds to the rdate parameter of the LABEL control statement.		
54	Accessibility (fa)	1	An a character which indicates the restrictions, if any, on who may have access to the information in this file. A blank means unlimited access. If fa is A, only the owner of the NOS written tape can access the file. If fa is any other character, all future accesses to the tape must specify this character as the fa parameter. File accessibility is not checked for system origin jobs.	Blank (unlimited access)	Yes, if a NOS written tape.
55-60	Block count	6	Must be zeros.		No
61-73	System code	13	13 a characters identifying the operating system that recorded this file. The tape is considered to have been written under NOS if the first 10 characters match the default.	KRONOS 2.1-nn (nn is the EST ordinal of the unit on which the file was written)	No
74-80	Reserved for future standardization	7	Must be spaces.		No

EOF1 — FIRST END-OF-FILE LABEL

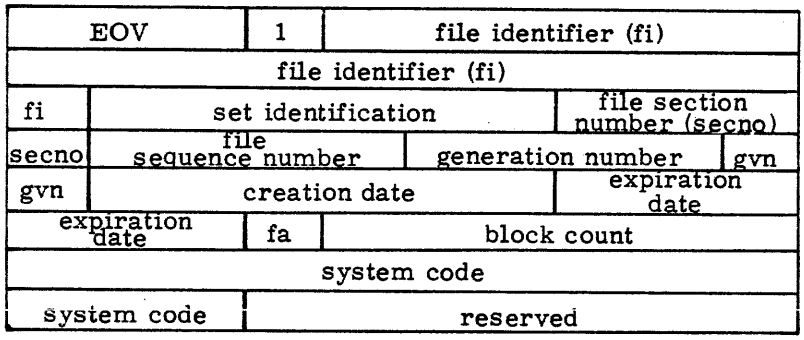
The end-of-file label is the last block of every file. It is the system end-of-information for the file. A single tape mark precedes EOF1. A double tape mark written after the EOF1 label marks the end of a multifile set.



Character Position	Field Name	Length (in characters)	Contents	Default	Checked on Read
1-3	Label identifier	3	Must be EOF.		Yes
4	Label number	1	Must be 1.		Yes
5-54	Same as corresponding fields in HDR1 (optional)	50	Same as the corresponding fields in HDR1.		Same as HDR1.
55-60	Block count	6	Six n characters specifying the number of data blocks between this label and the preceding HDR label group. This total does not include labels or tape marks.		Yes
61-80	Same as corresponding fields in HDR1 (optional)	20	Same as corresponding fields in HDR1.		Same as HDR1.

EOV1 — FIRST END-OF-VOLUME LABEL

The end-of-volume label is required only if the physical end-of-tape reflector is encountered before an EOF1 label is written or if a multifile set is continued on another volume. EOV1 is preceded by a single tape mark and followed by a double tape mark.



Character Position	Field Name	Length (in characters)	Contents	Default	Checked on Read
1-3	Label identifier	3	Must be EOV.		Yes
4	Label number	1	Must be 1.		Yes
5-54	Same as the corresponding fields in HDR1 (optional)	50	Same as the corresponding fields in HDR1.		Same as HDR1.
55-60	Block count	6	Six n characters specifying the number of data blocks between this label and the preceding HDR label group. This total does not include labels or tape marks.		Yes
61-80	Same as the corresponding fields in HDR1 (optional)	20	Same as the corresponding fields in HDR1.		Same as HDR1

OPTIONAL LABELS

Six types of optional labels are allowed. They are additional file header (HDR2-9), end-of-file (EOF2-9), end-of-volume (EOV2-9), user volume (UVLa), header (UHLa), and trailer (UTLa) labels.

HDR2-9 — ADDITIONAL FILE HEADER LABELS

HDR2-9 labels may immediately follow HDR1. Their format is:

Character Position	Field Name	Length (in characters)	Contents	Default Written
1-3	Label identifier	3	HDR	HDR
4	Label number	1	2-9	2-9
5-80		76		

Only the label identifier and the label number are checked on read.

EOF2-9 — ADDITIONAL END-OF-FILE LABELS

EOF2-9 labels may immediately follow EOF1. Their format is:

Character Position	Field Name	Length (in characters)	Contents	Default Written
1-3	Label identifier	3	EOF	EOF
4	Label number	1	2-9	2-9
5-80		76		

Only the label identifier and the label number are checked on read.

Refer to section 3 in volume 2 for a description of the use of EOV2 labels in conjunction with CLOSER, REWIND, and UNLOAD macros.

USER LABELS

User labels may immediately follow their associated system labels. Thus, user volume labels (UVLa) may follow VOL1, user header labels (UHLa) may follow the last HDRn label, and user trailer labels (UTLa) may follow the last EOVn or EOFn label. Their format is:

Character Position	Field Name	Length (in characters)	Contents	Default Written
1-3	Label identifier	3	UVL, UHL, or UTL.	UVL, UHL, or UTL.
4	Label number	1	Must be 1, 2, 3, 4, and so on, consecutively for UVL labels. For other labels, any a character.	
5-80	User option	76	Any a characters.	

Only the label identifier and the label number are checked on read. The system checks the number of user labels of a label type; a maximum of 64 is allowed.

CONTROL LANGUAGE (KCL) H

The control language (KCL) described in this appendix is the system control language available under NOS prior to the introduction of the CYBER Control Language (section 4). Support of KCL will be dropped in a future NOS release, and users are encouraged to convert their KCL procedures to CYBER Control Language.

The following paragraphs describe the various components and commands of KCL. If a KCL component is identical to its CCL counterpart, the user is referred to its description in section 4.

EXPRESSIONS

The expressions allowed are similar to FORTRAN expressions and may contain constants, operators, functions, and symbolic names.

OPERATORS

The arithmetic, relational, and logical operators are the same in the control language outlined in this appendix as they are in the CYBER Control language described in section 4. The only exception is the exclusive OR logical operator, .EOR.

FUNCTIONS

Two functions are provided for use in expressions specified with control language statements. The FILE function determines the status of any file assigned to the job. The NUM function determines if a specified parameter name has a numeric value. For complete information concerning format and use, refer to Control Language Functions in this section.

SYMBOLIC NAMES

Symbolic names are used to reference values pertaining to the job process. There are three categories of symbolic names, as follows:

- Symbolic names with fixed arithmetic values:

ARE	Arithmetic error.
BCO	Local batch origin.
CMM	Maximum CM field length (MFL setting).
CMN	Nominal CM field length (RFL setting).
CPE	CPU abort.
ECM	Maximum ECS field length.
ECN	Nominal ECS field length.

EIO	Remote batch origin.
FLE	File limit error.
FSE	Forced error.
MNE	Monitor call error.
ODE	Operator drop.
OKE	Operator kill drop.
ORE	Override error.
PEE	CPU parity error exit.
PPE	PPU abort.
PSE	Program stop error.
RRE	Rerun error.
SRE	SRU limit error.
SSE	Subsystem aborted.
SYE	System abort.
SYO	System origin.
TKE	Track limit error.
TLE	Time limit error.
TXO	Time-sharing origin.

- Symbolic names with variable arithmetic values which depend upon job state:

EF	Previous error flag.
EM	Current exit mode.
FL	Job field length.
OT	Job origin type.
R1	Contents of control register 1.
R2	Contents of control register 2.
R3	Contents of control register 3.
SS	Job subsystem. This particular symbolic name requires an equal sign. SS may be equivalenced to one of the following:

ACCESS
BASIC
BATCH
EXECUTE
FORTRAN
FTNTS
NULL
TRANACT(for TAF/TS only)†

†Special validation is necessary to use the library update and batch transaction functions of TRANACT. Refer to the LIMITS statement in section 6.

- Symbolic names with Boolean values:

F	False value.
FALSE	False value.
SWn	Setting (1 is on, 0 is off) of sense switch ($1 \leq n \leq 6$).
T	True value.
TRUE	True value.

EVALUATION OF EXPRESSIONS

The order of evaluation of expressions is:

1. Exponentiation
2. Multiplication, division
3. Addition, subtraction, negation
4. Relations
5. Complement
6. AND
7. Inclusive OR
8. Exclusive OR, equivalence

Nesting of expressions to any depth is allowed within a statement.

CONTROL LANGUAGE STATEMENTS

Control language statements are described in the following paragraphs. Separators and terminators must be used as shown in the statement formats. For descriptions of the SET and DISPLAY statements, refer to section 4.

GOTO STATEMENT

The GOTO statement transfers control to another location within the control statement file.

The statement format is:

GOTO,stmt.

stmt — Name of any control statement or a digit (0 through 9) followed by a maximum of 6 alphanumeric characters, terminated by a period.

Example 1

```
.
GOTO,1WX2.
.
.
.
.
1WX2,REQUEST(TAPE1)
.
.
```

Example 2

```
.
REQUEST(TAPE1)
.
.
.
GOTO,REQUEST.
.
REQUEST(TAPE2)
.
```

When stmt appears more than once in the control statement file, the stmt to be executed is the first occurrence of stmt from the beginning of the control statement file. Hence, in both of the previous examples, the REQUEST (TAPE1) statement is processed after the GOTO statement.

CALL STATEMENT

The CALL statement allows the user to insert a file consisting of a group of control statements (procedure file) at the specified position in the control statement stream. This file is merged, as specified on the CALL statement, with the current control statement record into a third record. This third record becomes the current control statement record. The remainder of the input file is then copied to the new control statement record. If the C option is exercised, the current control statement record is not used. Only the source file is used to generate a new control statement record. The C and S options are order-independent; the RENAME option, if present, must be last.

Lines within a procedure file may contain line numbers to make maintenance easier. Usually, the CALL statement strips off these line numbers before copying the procedure statements to the new control statement record. However, if a comma immediately follows the line number, the line number remains on the statement.

The statement format is:

CALL(lfn, C, S=ccc, RENAME(oldnam$_1$=newnam$_1$, oldnam$_2$=newnam$_2$, . . . , oldnam$_n$=newnam$_n$)

or

CALL(lfn, C, S=ccc(oldnam$_1$=newnam$_1$, oldnam$_2$=newnam$_2$, . . . , oldnam$_n$=newnam$_n$)

lfn — Procedure file name (refer to the description of procedure files in this section for further information). The system obtains lfn by:

- Searching for a local file, lfn
- Searching the system library for lfn
- Attempting to retrieve a working copy of an indirect access file

C — Replaces all of the control statement record after the CALL statement with lfn.

S=ccc — Sets next control statement to be processed to statement ccc. If S is not specified, the first statement in lfn is processed.

RENAME — Each occurrence of oldnam$_i$ is replaced with newnam$_i$ before the statement is entered into the statement file. As shown by the optional format, the word RENAME does not have to appear.

oldnam$_i$ — Old name; name of a file or statement label used in the specified procedure file.

newnam$_i$ — New name; name to replace oldnam$_i$.

If lfn is not properly formatted for a procedure file, the following message is issued.

lfn NOT A PROCEDURE FILE.

IF STATEMENT

The IF statement is used to evaluate an expression. If the conditions given in the expression are true, the dependent statement is processed. The expression is considered true if it is evaluated to a nonzero numeric value.

The statement format is:

IF(expression)stmt.

or

IF(SS=ssname)stmt.

expression	Any legal expression†
stmt	Any legal control statement.
ssname	Any legal SS subsystem name.

NOTE

A statement of the form IF(expression)CALL (lfn) is not recommended. Each time the IF statement is processed and the expression is true, the CALL statement is processed. This merges the called lfn with the current control statement stream and creates a copy of this procedure file each time.

Example 1:

```
IF(R2=R1.AND.R3)GOTO,REQUEST.
SET(EF=1)
    :
REQUEST(TAPE)
```

If the expression is true, the REQUEST control statement is executed; otherwise, the SET statement is executed.

† If a permanent file control statement is included in an IF statement, a password (if present) is not deleted in the dayfile.

Example 2:

```
IF(SS=BASIC)GOTO, 100.
SET(SS=BASIC)
    :
    :
100, OLD, BAS.
```

If the statement is true, the OLD control statement is processed; otherwise, the SET statement is processed.

CONTROL LANGUAGE FUNCTIONS

Control language functions are described in the following paragraphs. Separators and terminators must be used as shown in the function formats.

FILE FUNCTION

The FILE function is used to determine the status of any file assigned to the job and is used in conjunction with the SET, IF, and DISPLAY control language statements.

The format of the function is:

FILE(lfn, expression)

lfn — File name.

expression — Any legal expression; however, FILE expressions cannot include functions. In addition, FILE expressions use different symbolic names, as follows:

Symbolic names:

Names with values:

EQ	Equipment status table (EST) ordinal† (0 through 77_8).
ID	File ID (0 through 67_8).

File characteristics:

MS	File is on mass storage.
LK	File is locked.
OP	File is opened.
EX	Execute-only file.
AS	File is assigned to user's control point.

File types:

LO	Local.
PR	Print.
IN	Input.
PH	Punch.
LI	Library.
PM	Direct access permanent file.
PT	Primary.

†Contact installation personnel for a list of EST ordinals.

Device types:

CP	415 Card Punch.
CR	405 Card Reader.
DE	Extended core storage.
DI	844-21 Disk Storage Subsystem (half track).
DJ	844-4x Disk Storage Subsystem (half track).
DK	844-21 Disk Storage Subsystem (full track).
DL	844-4x Disk Storage Subsystem (full track).
DM	885 Disk Storage Subsystem (half track).
DP	Distributive data path to ECS.
DQ	885 Disk Storage Subsystem (full track).
LP	Any line printer.
LR	580-12 Line Printer.
LS	580-16 Line Printer.
LT	580-20 Line Printer.
MS	Mass storage.
MT	Magnetic tape drive (7-track).
NE	Null equipment.
NT	Magnetic tape drive (9-track).
TT	Time-sharing terminals.
NP	Host communications processor.

Examples:

```
SET(R1=FILE(TAPE,MT))
```

If TAPE is a file on a 7-track magnetic tape drive, R1 is set to 1; otherwise, it is set to zero.

```
IF(FILE(BETA,DI.AND.PM))GOTO,200.
```

If BETA is a file on an 844-21 Disk Storage Subsystem and it is a direct access permanent file, control skips to the statement at 200.

NUM FUNCTION

The NUM function is used to determine if the specified parameter name has a numeric value. It is used in conjunction with the SET, IF, and DISPLAY control language statements.

The format of the function is:

NUM(name)

name	Parameter name. If the name is numeric, the statement is true; otherwise, it is false.

Example:

If the following CALL statement is used to call procedure file A

```
CALL(A,RENAME(2XY=2,T=TAPE))
```

the IF statement in A

```
IF(NUM(2XY))GOTO,1S.
```

is evaluated as true, and control transfers to 1S.

However, the statement

```
IF(NUM(T))GOTO,1S.
```

is evaluated as false, and control passes to the next statement in A.

PROCEDURE FILES

Procedure files are source files consisting of control statements, control language statements, or both. The first statement of a procedure file may be the file name. If the first statement is the same as the file name used in the CALL statement, the first statement is ignored. Procedure files are activated by the CALL statement or by using the name of the procedure file, if the file is in the system.

Example 1:

The procedure file in this example is an indirect access file called COMPARE. This routine copies an input file and compares it with an existing direct access file. In the procedure file, these two files are called DUPL and MASTER. When the procedure file is inserted into the control statement record during job processing, the name of DUPL is changed to NEWFILE.

Original Input File

```
JOBAAA.
USER(EFD2501,PASS)
CHARGE(59,69N1)
CALL(COMPARE(DUPL=NEWFILE)
-EOR-
      .
    input file
    that is to
    be compared
      .
-EOI-
```

Procedure File COMPARE

```
COMPARE
COPYBR(,DUPL)
ATTACH(MASTER)
VFYLIB(MASTER,DUPL)
```

After the CALL control statement is processed, the control statement record is as follows:

```
JOBAAA.
USER(EFD2501,PASS)
CHARGE(59,69N1)
CALL(COMPARE(DUPL=NEWFILE))
COPYBR(,NEWFILE)
ATTACH(MASTER)
VFYLIB(MASTER,NEWFILE)
-EOR-
```

Example 2:

This is an example of nested calls. It illustrates the use of one procedure file to skip a specified number of files on a tape (contents of R1) and to copy source data to the tape. The other procedure file retrieves source data from the OPL (old program library) and calls the first procedure file to place that source data on the tape.

Input Deck

```
JOBAAA.
USER(USERNUM,PASSWRD,FAM1)
CHARGE(59,69N1)
ATTACH(OPL/UN=LIBRARY)
REQUEST(TAPE)
MODIFY(S,Z)/*EDIT,CPM
SET(R1=0)
CALL(PROC,RENAME(A=TAPE,B=SOURCE,2=2A,3=3A)
SET(R1=R1+1)
CALL(PROB)
-EOR-
```

Procedure File PROB

```
PROB
MODIFY(S=NEW,Z)/*EDIT,MTR
CALL(PROC,RENAME(A=TAPE,B=NEW)
RETURN,NEW.
```

Procedure File PROC

```
PROC
REWIND(A,B)
SET(R2=0)
2,IF(R1=R2)GOTO,3.
SKIPF(A)
SET(R2=R2+1)
GOTO,2.
3,COPYBF,B,A.
```

NOTE

On job initiation, the user's input file is a locked file. If the user wishes to call procedure files that write data on the input file, he should enter the RETURN (INPUT) control statement before attempting to write on INPUT. For further information, refer to Input File Control in section 3.

LINE PRINTER CARRIAGE CONTROL I

This appendix briefly describes the format and processing of print files.† It lists the carriage control for the programmable format control (PFC) and non-PFC 580 line printers.

PRINTED DATA

All data to be printed is in coded format in a print file within the print queue. The data consists of either 6-bit or 12-bit codes. Data recorded using the 6/12 display code set (refer to appendix A) should be converted to the 12-bit ASCII code set (refer to the FCOPY statement in section 7) before being routed to a line printer.

The system extracts data until an end-of-line occurs or until 137 characters are retrieved. End-of-line is 12 or more zero bits in the rightmost byte of a central memory word.

CARRIAGE CONTROL

The system interprets the first character in a line as the carriage control character†† and that character is not printed (table 1-I-1). The remainder of the line is then printed, except when the Q, R, S, or T carriage control characters are specified. The Q, R, S, T, and V format controls remain in effect until changed; all other carriage control characters must be supplied for each line they control. Line spacing is normally done in auto eject mode; that is, creases in the paper are skipped by the line printer's automatic line spacing mechanism if the paper is loaded properly. Auto eject mode must be turned off if the user wants to select format channels to advance printing from a position above the bottom of form to a position beyond the next top of form.

During the printing of each file, the system maintains a count of the number of lines printed or skipped for the file. If the number exceeds the limit for which the user is validated, printing of the file is terminated. The informative diagnostic LINE LIMIT EXCEEDED is printed. If a job's dayfile is part of the terminated print file, the dayfile is subsequently printed.

The installation can impose an implied page control by setting a certain number of default lines for each page. If less than the default number of lines is printed or skipped on a page, the line limit is still decremented by the default number of lines.

† To print a file in which the first character of each line is not a carriage control character, refer to the COPYSBF control statement in section 7.

†† The information in this appendix does not apply to remote batch line printers.

TABLE 1-I-1. CARRIAGE CONTROL CHARACTERS

Character	Action
SPACE	Single space.
1	Eject page before print.
0	Skip one line before print (double space).
-	Skip two lines before print (triple space).
+	Suppress space before print.
/	Suppress space after print.
2	Skip to last line of form before print.
Q	Clear auto eject;†† remainder of line is not printed.
R	Set auto eject; remainder of line is not printed.
S	Select 6 lines/inch; remainder of line is not printed.
T	Select 8 lines/inch; remainder of line is not printed.
V	Eject page before print on a 580 PFC printer, V loads a user-supplied PFC array (validated users only).
8	Skip to next punch in format channel 1 before print. †
7	Skip to next punch in format channel 2 before print. †
6	Skip to next punch in format channel 3 before print. †
5	Skip to next punch in format channel 4 before print. †
4	Skip to next punch in format channel 5 before print. †
3	Skip to next punch in format channel 6 before print. †
H	Skip to next punch in format channel 1 after print.
G	Skip to next punch in format channel 2 after print.
F	Skip to next punch in format channel 3 after print.
E	Skip to next punch in format channel 4 after print.
D	Skip to next punch in format channel 5 after print.
C	Skip to next punch in format channel 6 after print.

†No space after print. For all other control characters, a line feed is issued after print.

††The deselection of auto eject mode on a 580 line printer results in the deselection of 8 lines per inch, if previously selected.

CARRIAGE CONTROL USING FORMAT CHANNEL SELECTION

A programmer can use carriage control characters (table 1-I-1) that refer to channels (tracks) on a punched carriage control tape or in a programmable format control (PFC) array. The carriage control character beginning a print file line determines where on the printer form the line is printed.

After reading the carriage control character for a print line, the printer checks the format channel the character references. If a punch exists at that frame in that channel of the carriage tape or if a bit is set in that frame and channel of the PFC array, the printer prints the line. If not, the printer advances the carriage tape or PFC array and advances the printer paper until a punch or set bit is found in that channel. It then prints the line.

As listed in table 1-I-1, some carriage control characters name an action to be taken while others directly name a format channel. When specifying characters that name an action, the user indirectly names the format channel that performs the named action. To use the characters that directly name a format channel, the user needs to know the contents of the channel.

The format of the carriage control tape recommended for use on a 580 non-PFC line printer is listed in table 1-I-2. Lines 132 through 134 are identical to lines 0 through 2 because they overlap when the punched tape is glued together to form a continuous loop. Therefore, the user can disregard lines 132 through 134.

Selecting format channels on the carriage tape illustrated in figure 1-I-1 produces the following actions.

Carriage Control Character	Format Channel Selected	The line printer advances to:
8 or H	1	Line 0 or 66.
7 or G	2	First line of the next two-line group.
6 or F	3	First line of the next three-line group.
5 or E	4	First line of the next four-line group.
4 or D	5	First line of the next five-line group.
3 or C	6	Line 0.

If the numeric carriage control character is specified, the printer advances before printing. If the alphabetic character is specified, the printer advances after printing.

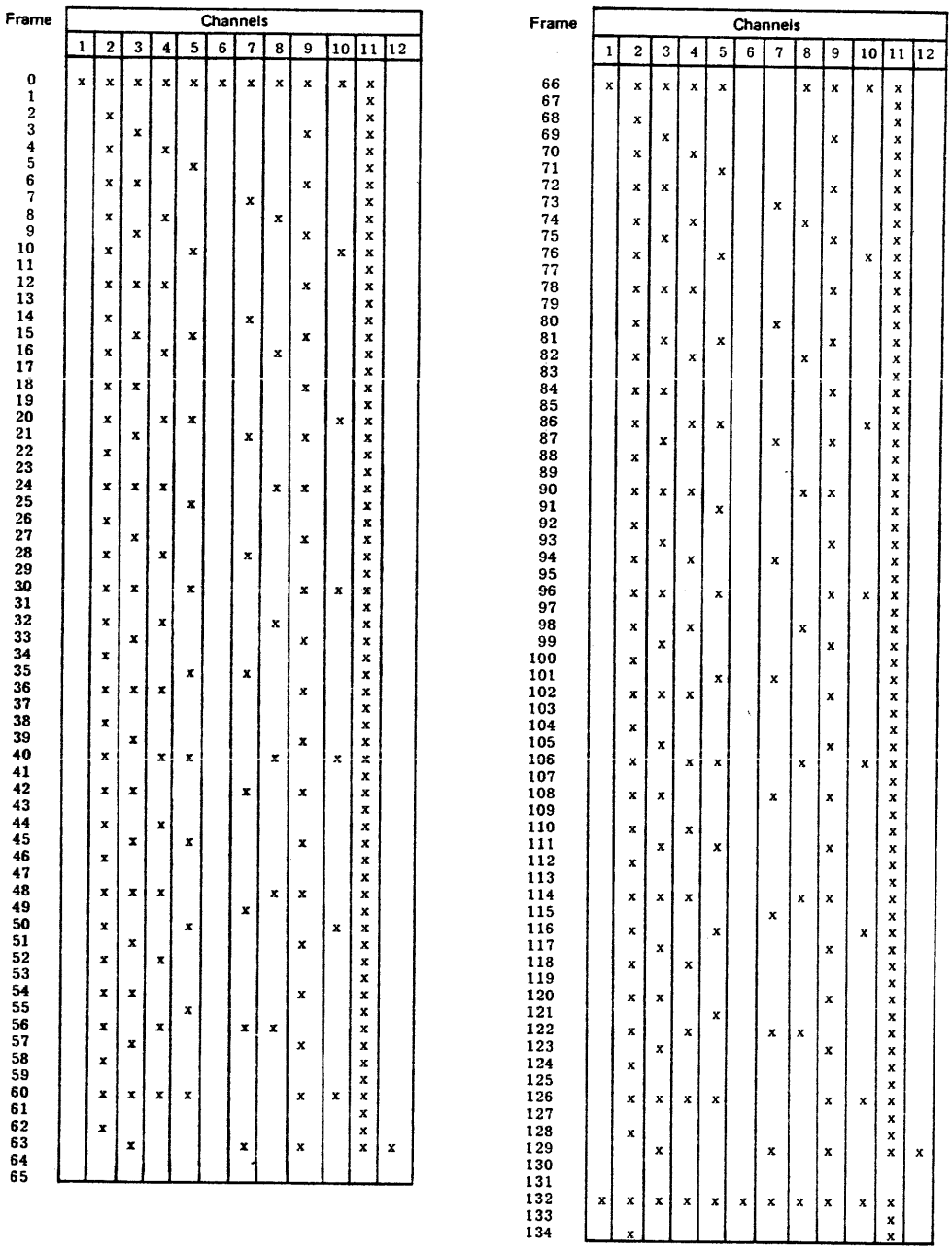


Figure 1-I-1. Carriage Control Tape Format

CARRIAGE CONTROL ARRAYS FOR 580 PFC PRINTERS

Line spacing on a 580 programmable format control (PFC) line printer is controlled by a PFC array that acts as a software version of the carriage control tape in non-PFC printers. The printer has two default PFC arrays, one for 6 lines-per-inch print density and the other for 8 lines-per-inch print density. The default arrays are listed in the System Maintenance Reference Manual. If validated (refer to the LIMITS statement, section 6), the user can include a PFC array in the print file to control its spacing until another PFC array or the end of the print file is reached. This user-supplied PFC array begins with the carriage control character V.

NOTE

The PFC array does not change the print density. Print density is selected by an S or T carriage control character.

Upon reading a V carriage control character, the line printer page ejects. If the printer is not a PFC printer, the rest of the line is ignored. If the user is not validated to use the V carriage control character, the print file terminates, and the user is informed by a message in his output file.

NOTE

The V carriage control character is ineffective when a print file is routed from a time-sharing job.

PFC ARRAY SYNTAX

The carriage control character V is in column 1 of the line. The character in column 2 determines which PFC array is changed. Its legal values are the following.

- 6 Six-line-per inch spacing. The entire array specification is on this line.
- 8 Eight-line-per-inch spacing. The entire array specification is on this line.
- C Eight-line-per-inch spacing. The array specification is continued on a second line. The second line begins in column 3. Columns 1 and 2 are ignored.

The PFC array specifications start in column 3. They are the alphabetic characters A through L, O, and blank and have the following significance.

A	Specifies top of forms code. This must be the first character in the array.
B through K	Specifies format channels 2 through 11,† respectively.
L	Specifies bottom of forms code.
O	Signals the end of the array specification. This must be the last character in the array. It has no effect on printing.
Blank	No channel specified.

† The A character refers to channel 1, and L refers to channel 12. In the maximum length array specification (132 for six lines-per-inch and 176 for eight lines-per-inch), the J character can be specified only in the last position (immediately before the O).

NOTE

The user must specify each channel referenced in the print file in the PFC array. A channel specification can be repeated.

A six-line-per-inch array specification may be a maximum of 132 characters plus the array terminator. An eight-line-per-inch array specification may be 176 characters plus the array terminator.

If the array contains an illegal character, the array line is printed and the print file is terminated. Other invalid arrays are ignored and the file is printed using the carriage control array previously loaded into the printer.

Examples:

The following arrays are invalid.

V6BCD O	Array does not begin with an A.
VBA C DEO	Second character is not 6, 8, or C.
V8ABWCO	Contains an illegal character (W).

The following example uses a PFC array to print a short special form.

A programmer wants to print on the top, fifth, and bottom lines of an eight-line form (eight lines-per-inch print density). He selects the print density with the T carriage control character and then loads the following PFC array.

```
Columns:  1 2 3 4 5 6 7 8 9 10 11
          V 8 A       B     L  O
```

The eight lines-per-inch carriage control array is changed as follows (x denotes a bit set):

	Channels											
	1	2	3	4	5	6	7	8	9	10	11	12
1	x											
2												
3												
4												
5		x										
6												
7												
8												x

Because only 3 of the 12 format channels are specified in the array, the other 9 channels (in the eight lines-per-inch array) remain unchanged from their last setting.

After loading the array, output lines beginning with an 8 (format channel 1) are printed at the top of form, lines beginning with a 7 (format channel 2) are printed at the fifth line, and lines beginning with a 2 (format channel 12) are printed at the bottom of the form.

INDEX

COMMENT SHEET

MANUAL TITLE: CDC NOS Version 1 Reference Manual, Volume 1

PUBLICATION NO.: 60435400 **REVISION:** J

NAME: _____

COMPANY: _____

STREET ADDRESS: _____

CITY: _____ **STATE:** _____ **ZIP CODE:** _____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

CUT ALONG LINE

AA3419 REV. 4/79 PRINTED IN U.S.A.

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

FOLD ON BOTTOM LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55112

CUT ALONG LINE

FOLD

FOLD