60435400

CONTROL DATA
CORPORATION

NOS VERSION 1
REFERENCE MANUAL

Volume 1 of 2

CDC® COMPUTER SYSTEMS:
CYBER 170 MODELS 172, 173, 174, 175
CYBER 70 MODELS 71, 72, 73, 74
6000 SERIES

ALPHABETICAL LIST OF CONTROL STATEMENTS †

ACCOUNT	1-6-2	GOTO	1-4-4	RBR	1-9-4
ALGOL	1-11-3	GPSS	84003900	REDUCE	60429800
APEX	76070000	GTR	1-7-17	RENAME	1-7-26
APPEND	1-8-5	IF	1-4-7	REPLACE	1-8-15
ASCII	1-4-12	ISF	60435700	REQUEST	1-7-27, 10-17
ASSIGN††††	1-7-2, 10-11	ITEMIZE	60495600	RERUN	1-6-14
ATTACH	1-8-6	JOB	1-5-4	RESEQ	1-7-29
BASIC	1-11-6	KRONREF	1-14-10	RESOURC	1-6-15
BKSP	1-7-3	LABEL	1-10-13	RESTART	1-12-2
BLANK	1-10-12	LBC	1-9-2	RETURN	1-7-29
CALL	1-4-5	LDI	1-6-7	REWIND	1-7-30
CATALOG	1-7-4	LDSET	60429800	RFL	1-6-17
CATLIST	1-8-8	LENGTH	1-6-8	ROLLOUT	1-6-18
CHANGE	1-8-10	LIBEDIT	1-7-18	ROUTE	1-7-31
CHARGE	1-6-2	LIBGEN	1-7-19	RTIME	1-6-18
	60435700	LIBLOAD	60429800	SATISFY	60429800
CKP	1-12-1	LIBRARY	60429800	SAVE	1-8-16
CLEAR	1-7-7	LIMITS	1-6-8	SCRSIM†††	60435700
COBOL	1-11-8		60435700	SET	1-4-6
COBOL5	1-11-11	LISTLB	1-10-16	SETASL	1-6-18
COMMENT	1-6-3	LIST80	1-7-20	SETCORE	1-6-19
COMMON	1-7-7	LOAD	60429800	SETID	1-7-34
COMPASS	60445300	LOC	1-9-3	SETJSL	1-6-19
CONVERT	1-7-7	LOCK	1-7-21	SETPR	1-6-20
COPY	1-7-8	LO72	1-7-21	SETTL	1-6-20
COPYBF	1-7-9	MAP	60429800	SKIPEI	1-7-35
COPYBR	1-7-10	MFL	1-6-11	SKIPF	1-7-35
COPYCF	1-7-10	MODE	1-6-11	SKIPFB	1-7-35
COPYCR	1-7-11	MODIFY	1-14-2	SKIPR	1-7-36
COPYEI	1-7-12	MODVAL††††	60435700	SLOAD	60429800
COPYL	60495600	NEW	1-7-24	SORT	1-7-36
COPYSBF	1-7-12	NOEXIT	1-6-13	SORTMRG	1-11-23
COPYX	1-7-13	NOGO	60429800	STAGE	1-7-38
CSET	1-4-12	NORERUN	1-6-13	STIME	1-6-20
CTIME	1-6-3	OFFSW	1-6-13	SUBMIT	1-6-21
DAYFILE	1-6-3	OLD	1-8-12	SUI††	1-6-25
DEFINE	1-8-11	ONEXIT	1-6-13	SUMMARY	1-6-25
DISPLAY	1-4-6	ONSW	1-6-14	SWITCH	1-6-26
DISPOSE	1-7-14	OPLEDIT	1-14-4	SYSEDIT†††	60435700
DMD	1-9-2		60450100	TDUMP	1-7-39
DMP	1-9-1	OUT	1-7-24	UNLOAD	1-7-40
DOCMENT	1-7-15	PACK	1-7-25	UNLOCK	1-7-40
DSDI	60435700	PACKNAM	1-8-13	UPDATE	1-14-6
EDIT	1-14-1	PARITY	1-4-12	UPMOD	1-14-9
	60436100	PASSWOR	1-6-14	USECPU	1-6-26
ENQUIRE	1-6-4		60435700	USER	1-6-27
EVICT	1-7-16	PBC	1-9-4	VALNET	60435700
EXECUTE	60429800	PERMIT	1-8-14	VERIFY	1-7-41
EXIT	1-6-7			VFYLIB	1-7-42
FAMILY††	60435700	PRIMARY	1-7-26	VSN	1-10-18
FORMAT†††††	60435700	PROFILE††††	60435700	WBR	1-9-5
FTN	1-11-17	PURGALL	1-8-14	WRITEF	1-7-42
GET	1-8-12	PURGE	1-8-15	WRITER	1-7-42

† Reference to a page number indicates the statement is described in this manual; a manual publication number means the statement is described in that manual.

†† For system origin jobs only.

††† For system origin jobs or users with system origin privileges and DEBUG mode on on at the console.

†††† Some features of this statement require system origin privileges.

††††† For system origin jobs or users with system origin privileges and ENGINEERING mode on at the console.

60435400

CONTROL DATA
CORPORATION

NOS VERSION 1
REFERENCE MANUAL

Volume 1 of 2

CDC® COMPUTER SYSTEMS:
CYBER 170 MODELS 172, 173, 174, 175
CYBER 70 MODELS 71, 72, 73, 74
6000 SERIES

REVISION RECORD

REVISION	DESCRIPTION
A (6-17-75)	Manual released.
B (3-8-76)	Revised to reflect NOS 1.1 as well as technical and literary corrections. New features include support of memory increments to 262K on CDC CYBER 170 Series Systems, 844-41 Disk Storage Subsystem, multimainframe, 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 (12-3-76)	Revised to reflect NOS 1.2 at PSR level 439. New features include revised field length control, 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 COBOL5 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.

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.

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
Front Cover	-
Control Statements	C
Title Page	-
ii	C
iii	C
iv	C
v	C
vi	C
vii	C
viii	Blank
ix	C
x	C
xi	C
xii	C
xiii	C
xiv	C
1-1-1	C
1-1-2	C
1-1-3	C
1-1-4	C
1-2-1	C
1-2-2	C
1-2-3	C
1-2-4	C
1-2-5	C
1-2-6	C
1-2-7	C
1-2-8	C
1-2-9	C
1-2-10	B
1-2-11	B
1-2-12	B
1-2-13	B
1-2-14	C
1-3-1	C
1-3-2	C
1-3-3	C
1-3-4	C
1-3-5	C
1-3-6	C
1-3-7	C
1-3-8	C
1-3-9	C
1-3-10	C
1-3-11	C
1-3-12	C
1-3-13	C
1-3-14	C
1-3-15	C
1-3-16	C
1-4-1	B
1-4-2	B
1-4-3	C
1-4-4	B
1-4-5	C
1-4-6	A
1-4-7	B
1-4-8	B

PAGE	REV
1-4-9	C
1-4-10	C
1-4-11	C
1-4-12	A
1-5-1	B
1-5-2	C
1-5-3	C
1-5-4	C
1-5-5	C
1-5-6	C
1-5-7	A
1-5-8	C
1-6-1	C
1-6-2	C
1-6-3	C
1-6-4	C
1-6-5	C
1-6-6	C
1-6-7	C
1-6-8	C
1-6-9	C
1-6-10	C
1-6-11	C
1-6-12	C
1-6-13	C
1-6-14	C
1-6-15	C
1-6-16	C
1-6-17	C
1-6-18	C
1-6-19	C
1-6-20	C
1-6-21	C
1-6-22	C
1-6-23	C
1-6-24	C
1-6-25	C
1-6-26	C
1-6-27	C
1-6-28	C
1-7-1	C
1-7-2	C
1-7-3	A
1-7-4	C
1-7-5	C
1-7-6	A
1-7-7	B
1-7-8	B
1-7-9	C
1-7-10	C
1-7-11	C
1-7-12	C
1-7-13	C
1-7-14	C
1-7-15	C
1-7-16	C
1-7-17	C
1-7-18	C
1-7-19	C

PAGE	REV
1-7-20	C
1-7-21	C
1-7-22	C
1-7-23	C
1-7-24	C
1-7-25	C
1-7-26	C
1-7-27	C
1-7-28	C
1-7-29	C
1-7-30	C
1-7-31	C
1-7-32	C
1-7-33	C
1-7-34	C
1-7-35	C
1-7-36	C
1-7-37	C
1-7-38	C
1-7-39	C
1-7-40	C
1-7-41	C
1-7-42	C
1-8-1	C
1-8-2	C
1-8-3	A
1-8-4	C
1-8-5	A
1-8-6	C
1-8-7	C
1-8-8	C
1-8-9	C
1-8-10	C
1-8-11	C
1-8-12	C
1-8-13	C
1-8-14	C
1-8-15	C
1-8-16	C
1-9-1	C
1-9-2	C
1-9-3	A
1-9-4	A
1-9-5	C
1-10-1	A
1-10-2	C
1-10-3	C
1-10-4	C
1-10-5	C
1-10-6	C
1-10-7	C
1-10-8	C
1-10-9	C
1-10-10	A
1-10-11	C
1-10-12	B
1-10-13	C
1-10-14	C
1-10-15	C

PAGE	REV
1-10-16	C
1-10-17	C
1-10-18	C
1-10-19	C
1-10-20	C
1-10-21	C
1-10-22	C
1-10-23	C
1-10-24	C
1-10-25	C
1-10-26	C
1-10-27	C
1-10-28	C
1-10-29	C
1-11-1	C
1-11-2	C
1-11-3	C
1-11-4	C
1-11-5	C
1-11-6	C
1-11-7	C
1-11-8	C
1-11-9	C
1-11-10	C
1-11-11	C
1-11-12	C
1-11-13	C
1-11-14	C
1-11-15	C
1-11-16	C
1-11-17	C
1-11-18	C
1-11-19	C
1-11-20	C
1-11-21	C
1-11-22	C
1-11-23	C
1-11-24	C
1-12-1	C
1-12-2	A
1-12-3	A
1-13-1	A
1-13-2	A
1-13-3	A
1-13-4	A
1-13-5	A
1-13-6	A
1-13-7	A
1-13-8	A
1-13-9	A
1-13-10	A
1-13-11	A
1-13-12	A
1-13-13	A
1-14-1	B
1-14-2	B
1-14-3	B
1-14-4	B
1-14-5	B

PAGE	REV
1-14-6	B
1-14-7	B
1-14-8	B
1-14-9	B
1-14-10	B
1-A-1	C
1-A-2	C
1-A-3	C
1-A-4	B
1-A-5	A
1-A-6	A
1-A-7	A
1-A-8	A
1-B-1	A
1-B-2	C
1-B-3	C
1-B-4	C
1-B-5	C
1-B-6	C
1-B-7	C
1-B-8	C
1-B-9	C
1-B-10	C
1-B-11	C
1-B-12	C
1-B-13	C
1-B-14	C
1-B-15	C
1-B-16	C
1-B-17	C
1-B-18	C
1-B-19	C
1-B-20	C
1-B-21	C
1-B-22	C
1-B-23	C
1-B-24	C
1-B-25	C
1-B-26	C
1-B-27	C
1-B-28	C
1-B-29	C
1-B-30	C
1-B-31	C
1-B-32	C
1-B-33	C
1-B-34	C
1-B-35	C
1-B-36	C
1-B-37	C
1-C-1	C
1-C-2	B
1-C-3	A
1-C-4	A
1-C-5	C
1-C-6	C
1-C-7	A
1-C-8	A
1-C-9	A

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
1-C-10	A								
1-C-11	C								
1-C-12	C								
1-C-13	C								
1-C-14	C								
1-C-15	C								
1-C-16	C								
1-C-17	C								
1-C-18	C								
1-D-1	A								
1-D-2	B								
1-D-3	B								
1-D-4	C								
1-E-1	C								
1-F-1	C								
1-F-2	A								
1-F-3	A								
1-F-4	A								
1-F-5	C								
1-F-6	A								
1-G-1	A								
1-G-2	A								
1-G-3	A								
1-G-4	A								
1-G-5	A								
1-G-6	A								
1-G-7	C								
1-G-8	A								
1-G-9	A								
1-G-10	A								
1-G-11	A								
1-G-12	A								
1-G-13	A								
1-G-14	A								
1-G-15	A								
1-G-16	A								
1-G-17	A								
1-G-18	A								
1-G-19	A								
1-G-20	C								
1-H-1	C								
1-H-2	C								
1-H-3	C								
1-H-4	C								
1-H-5	C								
Index-1	C								
Index-2	C								
Index-3	C								
Index-4	C								
Index-5	C								
Index-6	C								
Index-7	C								
Index-8	C								
Index-9	C								
Index-10	C								
Index-11	C								
Cmt Sheet	C								
Return Env	-								
Back Cover	-								

PREFACE

The Network Operating System (NOS) was developed by Control Data Corporation to provide network capabilities for time-sharing and transaction processing, in addition to local and remote batch processing, on CONTROL DATA® CYBER 170 Series, Models 172, 173, 174, and 175 Computer Systems, CONTROL DATA® CYBER 70 Series, Models 71, 72, 73, and 74 Computer Systems, and CONTROL DATA® 6000 Series Computer Systems.

This manual describes the external features of NOS 1.2 for the batch user. Information in this manual should be useful to those who use the programs and utilities supplied with the system and those who wish to write their own. The manual is contained in two volumes to separate information pertaining primarily to the applications programmer from that pertaining to the applications COMPASS programmer.

Volume 1 (publication no. 60435400) contains information for the applications programmer. This includes general information about files, job flow and execution, control statement processing, and an extensive discussion on control statements.

Volume 2 (publication no. 60445300) contains information for those who write system or assembly language programs for use with NOS. It is primarily intended for the applications COMPASS programmer; however, several portions 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.

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 this manual.

This manual does not contain a description of NOS system operation, detailed descriptions of the software product set available under NOS, or descriptions of the time-sharing commands.

The user is assumed to be familiar with CDC computer systems and with operating systems in general. For further information concerning CDC CYBER 170, CDC CYBER 70, and 6000 Series Computer Systems, the NOS time-sharing system, and the products supported by NOS, consult the following manuals.

Control Data Publication	Publication No.
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
6400/6500/6600 Computer Systems Reference Manual	60100000

Control Data Publication	Publication No.
NOS General Information Manual	60435900
NOS Installation Handbook	60435700
NOS Operator's Guide	60435600
NOS Application Programmer's Instant	60436000
NOS System Programmer's Instant	60449200
NOS Time-Sharing User's Reference Manual	60435500
NOS Time-Sharing User's Guide	60436400
NOS Terminal User's Instant Manual	60435800
NOS Text Editor Reference Manual	60436100
NOS Export/Import Reference Manual	60436200
TRANEX 1 Reference Manual	60407900
TRANEX 1/TAF 1 User's Guide	60436500
TAF 1 Reference Manual	60453000
TAF 1 Data Manager Reference Manual	60453100
NOS Modify Reference Manual	60450100
NOS Modify Instant	60450200
Update Reference Manual	60449900
NAM Reference Manual	60499500
Network Definition Languages Reference Manual	60480000
Network Administrator and Operator Facility Reference Manual	60480100
BASIC 3 Reference Manual	19983900
APL *CYBER Reference Manual	19980400
APL 2.0 Reference Manual	60454000
FORTRAN Extended 4 Reference Manual	60497800
COBOL 4 Reference Manual	60496800
COBOL 5 Reference Manual	60497100
ALGOL 4 Reference Manual	60496600
Sort/Merge 4 Reference Manual	60497500
CYBER Record Manager 1 Reference Manual	60495700
CYBER Loader 1 Reference Manual	60429800
FORM 1 Reference Manual	60496200
COMPASS 3 Reference Manual	60492600
SYMPL 1 Reference Manual	60496400
CDCS 1 Reference Manual	60498700
Data Base Utilities 1 Reference Manual	60498800
Query Update 2 Reference Manual	60384900

Control Data Publication	Publication No.
DDL 1 Reference Manual	60359000
Query Update 3 Reference Manual	60498300
DDL 2 Reference Manual	60498400
SIMSCRIPT 3 Reference Manual	60358500
SIMULA 1 Reference Manual	60234800
APEX III 1 Reference Manual	76070000
APT IV 2 Reference Manual	60499300
LCGT/IGS 1 Reference Manual	17322800
GPSS V/6000 1 General Information Manual	84003900
PERT/Time 1 Reference Manual	60133600
Total Universal 1 Reference Manual	76070300
8-Bit Subroutines 1 Reference Manual	60495500
Math Science Library 1 Reference Manual	60327500
Common Utilities Reference Manual	60495600
Application Installation Handbook	76071100
On-Line Maintenance Software Reference Manual	60436600
RBF Reference Manual	60499600

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

VOLUME 1

SECTION 1 SYSTEM DESCRIPTION 1-1-1
Central Processor Unit 1-1-1
Central Memory 1-1-1
  Central Memory Resident 1-1-2
  Extended Core Storage 1-1-2
Peripheral Processor Units 1-1-2
Peripheral Hardware 1-1-4
System Software 1-1-4
User Programs 1-1-4

SECTION 2 FILES 1-2-1
Logical/Physical File Structure 1-2-1
  Mass Storage Device File Structure 1-2-2
  Magnetic Tape File Structure 1-2-2
  Punch File Structure 1-2-2
File Types 1-2-3
  Queue Files 1-2-3
    Input Files (INFT) 1-2-3
    Rollout Files (ROFT) 1-2-3
    Timed/Event Rollout Files (TEFT) 1-2-3
    Print Files (PRFT) 1-2-4
    Punch Files (PHFT) 1-2-4
  Special Files 1-2-4
    Local Files (LOFT) 1-2-4
    Direct Access Permanent File (PMFT) 1-2-5
    Library Files (LIFT) 1-2-5
    System Files (SYFT) 1-2-6
    Primary Terminal Files (PTFT) 1-2-6
Queued File Management 1-2-6
Reserved File Names 1-2-6
Permanent Files 1-2-7
Device Residence 1-2-8
Accessing Files 1-2-9
  Reading Files 1-2-9
  Writing Files 1-2-11
Libraries 1-2-12

SECTION 3 JOB FLOW AND EXECUTION 1-3-1
Job Initiation 1-3-6
Job Origin Types 1-3-6
Job Names 1-3-6
  System Origin Type (SYOT) Job Name Format 1-3-6
  Batch Origin Type (BCOT) Job Name Format 1-3-7
  Time-Sharing and Export/Import (TXOT and EIOT) Job Name Format 1-3-7
Validation 1-3-7
Accounting 1-3-7
Job Scheduling 1-3-8

	Job Control	1-3-8
	Field Length Control	1-3-8
	Input File Control	1-3-12
	Time Limit Control	1-3-12
	SRU Limit Control	1-3-12
	Rollout Control	1-3-13
	Error Control	1-3-13
	Security Control	1-3-15
	Job Completion	1-3-15
SECTION 4	CONTROL LANGUAGE	1-4-1
	Expressions	1-4-1
	Constants	1-4-1
	Arithmetic Operators	1-4-1
	Relational Operators	1-4-2
	Boolean Operators	1-4-2
	Functions	1-4-2
	Symbolic Names	1-4-2
	Evaluation of Expressions	1-4-4
	Control Language Statements	1-4-4
	GOTO Statement	1-4-4
	CALL Statement	1-4-5
	DISPLAY Statement	1-4-6
	SET Statement	1-4-6
	IF Statement	1-4-7
	Control Language Functions	1-4-8
	FILE Function	1-4-8
	NUM Function	1-4-9
	Procedure Files	1-4-10
	Time-Sharing Commands	1-4-12
	ASCII Statement	1-4-12
	CSET Statement	1-4-12
	PARITY Statement	1-4-12
SECTION 5	CONTROL STATEMENT PROCESSING	1-5-1
	Control Statement Format	1-5-1
	Job Statement Format	1-5-4
	Control Statement Processing Flow	1-5-6
	Exit Processing	1-5-8
SECTION 6	JOB CONTROL CONTROL STATEMENTS	1-6-1
	ACCOUNT	1-6-2
	CHARGE Statement	1-6-2
	COMMENT Statement	1-6-3
	CTIME Statement	1-6-3
	DAYFILE Statement	1-6-3
	ENQUIRE Statement	1-6-4
	EXIT Statement	1-6-7
	LDI Statement	1-6-7
	LENGTH Statement	1-6-8
	LIMITS Statement	1-6-8
	MFL Statement	1-6-11
	MODE Statement	1-6-11
	NOEXIT Statement	1-6-13
	NORERUN Statement	1-6-13
	OFFSW Statement	1-6-13
	ONEXIT Statement	1-6-13
	ONSW Statement	1-6-14

PASSWOR Statement 1-6-14
RERUN Statement 1-6-14
RESOURC Statement 1-6-15
RFL Statement 1-6-17
ROLLOUT Statement 1-6-18
RTIME Statement 1-6-18
SETASL Statement 1-6-18
SETCORE Statement 1-6-19
SETJSL Statement 1-6-19
SETPR Statement 1-6-20
SETTL Statement 1-6-20
STIME Statement 1-6-20
SUBMIT Statement 1-6-21
SUI Statement 1-6-25
SUMMARY Statement 1-6-25
SWITCH Statement 1-6-26
USECPU Statement 1-6-26
USER Statement 1-6-27

SECTION 7 FILE MANAGEMENT CONTROL STATEMENTS 1-7-1
ASSIGN Statement 1-7-2
BKSP Statement 1-7-3
CATALOG Statement 1-7-4
CLEAR Statement 1-7-7
COMMON Statement 1-7-7
CONVERT Statement 1-7-7
COPY Statement 1-7-8
COPYBF Statement 1-7-9
COPYBR Statement 1-7-10
COPYCF Statement 1-7-10
COPYCR Statement 1-7-11
COPYEI Statement 1-7-12
COPYSBF Statement 1-7-12
COPYX Statement 1-7-13
DISPOSE Statement 1-7-14
DOCMENT Statement 1-7-15
EVICT Statement 1-7-16
GTR Statement 1-7-17
LIBEDIT Statement 1-7-18
LIBGEN Statement 1-7-19
LIST80 Statement 1-7-20
LOCK Statement 1-7-21
LO72 Statement 1-7-21
NEW Statement 1-7-24
OUT Statement 1-7-24
PACK Statement 1-7-25
PRIMARY Statement 1-7-26
RENAME Statement 1-7-26
REQUEST Statement 1-7-27
RESEQ Statement 1-7-29
RETURN Statement 1-7-29
REWIND Statement 1-7-30
ROUTE Statement 1-7-31
SETID Statement 1-7-34
SKIPEI Statement 1-7-35
SKIPF Statement 1-7-35
SKIPFB Statement 1-7-35

	SKIPR Statement	1-7-36
	SORT Statement	1-7-36
	STAGE Statement	1-7-38
	TDUMP Statement	1-7-39
	UNLOAD Statement	1-7-40
	UNLOCK Statement	1-7-40
	VERIFY Statement	1-7-41
	VFYLIB Statement	1-7-42
	WRITEF Statement	1-7-42
	WRITER Statement	1-7-42
SECTION 8	PERMANENT FILE CONTROL STATEMENTS	1-8-1
	APPEND Statement	1-8-5
	ATTACH Statement	1-8-6
	CATLIST Statement	1-8-8
	CHANGE Statement	1-8-10
	DEFINE Statement	1-8-11
	GET Statement	1-8-12
	OLD Statement	1-8-12
	PACKNAM Statement	1-8-13
	PERMIT Statement	1-8-14
	PURGALL Statement	1-8-14
	PURGE Statement	1-8-15
	REPLACE Statement	1-8-15
	SAVE Statement	1-8-16
SECTION 9	LOAD/DUMP CENTRAL MEMORY UTILITY CONTROL STATEMENTS	1-9-1
	DMP Statement	1-9-1
	DMD Statement	1-9-2
	LBC Statement	1-9-2
	LOC Statement	1-9-3
	PBC Statement	1-9-4
	RBR Statement	1-9-4
	WBR Statement	1-9-5
SECTION 10	TAPE MANAGEMENT	1-10-1
	ASSIGN Statement	1-10-11
	BLANK Statement	1-10-12
	LABEL Statement	1-10-13
	LISTLB Statement	1-10-16
	REQUEST Statement	1-10-17
	VSN Statement	1-10-18
	Magnetic Tape Formats	1-10-21
	Data Formats	1-10-21
	End-of-Tape/End-of-Reel Conditions	1-10-29
SECTION 11	PRODUCT SET CONTROL STATEMENTS	1-11-1
	User Libraries	1-11-1
	Control Statement Formats	1-11-2
	ALGOL Statement	1-11-3
	BASIC Statement	1-11-6
	COBOL Statement	1-11-8
	COBOL5 Statement	1-11-11
	FTN Statement	1-11-17
	SORTMRG Statement	1-11-23

SECTION 12	CHECKPOINT/RESTART	1-12-1
	CKP Statement	1-12-1
	RESTART Statement	1-12-2
SECTION 13	DEBUGGING AIDS	1-13-1
	Central Memory Dumps	1-13-1
	Generating Meaningful Dumps	1-13-2
	Reading CM Dumps	1-13-3
SECTION 14	SYSTEM UTILITY CONTROL STATEMENTS	1-14-1
	EDIT Statement	1-14-1
	MODIFY Statement	1-14-2
	OPLEDIT Statement	1-14-4
	UPDATE Statement	1-14-6
	UPMOD Statement	1-14-9
	KRONREF Statement	1-14-10

APPENDIXES

APPENDIX A	CHARACTER SETS	1-A-1
APPENDIX B	DAYFILE MESSAGES	1-B-1
APPENDIX C	LIBEDIT	1-C-1
APPENDIX D	JOB OUTPUT INFORMATION	1-D-1
APPENDIX E	PERMANENT FILE DEVICE STATISTICS	1-E-1
APPENDIX F	CARD FORMAT AND CONVERSION PROBLEMS	1-F 1
APPENDIX G	TAPE LABELS	1-G-1
APPENDIX H	GLOSSARY	1-H-1

INDEX

FIGURES

1-1-1	Central Memory Layout	1-1-3
1-2-1	Sample Card File Structure	1-2-2
1-2-2	Sample Random Access File Format	1-2-10
1-2-3	Modified Sample Random Access File	1-2-13
1-3-1	Basic Job Deck	1-3-1
1-3-2	COMPASS Assemble and Execute Deck	1-3-2
1-3-3	COMPASS Assemble, Execute, and Punch Binary Deck	1-3-3
1-3-4	FORTRAN Compile and Execute Deck	1-3-4
1-3-5	FORTRAN Load and Run Deck	1-3-5
1-3-6	Field Length Control Statement	1-3-10
1-5-1	Control Statement Processing Flow	1-5-7
1-7-1	Sample Page of Catalog of SYSTEM	1-7-5
1-13-1	Exchange Package	1-13-1
1-13-2	Main Program of Main Overlay (0,0)	1-13-5
1-13-2	Function Subroutine of Main Overlay (0,0)	1-13-6
1-13-4	Subroutine of Main Overlay (0,0)	1-13-6

1-13-5 Main Program of Primary Overlay (1, 0) 1-13-7
1-13-6 Loader Map of Main Overlay (0, 0) 1-13-8
1-13-7 Loader Map of Primary Overlay (1, 0) 1-13-11
1-13-8 Program Output 1-13-11
1-13-9 Exchange Package Dump 1-13-12
1-13-10 Central Memory Dump 1-13-12

TABLES

1-8-1 Combinations of Multiple Access 1-8-7
1-11-1 Principal Products Supported by NOS 1-11-1

VOLUME 2

SECTION 1 INTRODUCTION 2-1-1

SECTION 2 PROGRAM/SYSTEM COMMUNICATION 2-2-1

SECTION 3 FILE CREATION AND INPUT/OUTPUT 2-3-1

SECTION 4 LOCAL FILE MANAGER 2-4-1

SECTION 5 PERMANENT FILE MANAGER 2-5-1

SECTION 6 CONTROL POINT MANAGER 2-6-1

SECTION 7 QUEUE FILE MANAGER 2-7-1

SECTION 8 QUEUE DUMP/LOAD PROCESSOR 2-8-1

SECTION 9 SYSTEM FILE MANAGER 2-9-1

SECTION 10 JOB CONTROL 2-10-1

SECTION 11 SYSTEM/LOADER REQUESTS 2-11-1

SECTION 12 PROGRAM WRITING TECHNIQUES 2-12-1

APPENDIXES

APPENDIX A CPU COMMON DECKS 2-A-1

APPENDIX B EXAMPLES OF RANDOM I/O 2-B-1

APPENDIX C CODING SPECIFICATIONS 2-C-1

APPENDIX D PROGRAM EXAMPLE 2-D-1

APPENDIX E SPECIAL USER INFORMATION 2-E-1

APPENDIX F SPECIAL ENTRY POINTS 2-F-1

APPENDIX G BINARY FORMATS 2-G-1

APPENDIX H COMPASS CONTROL STATEMENT 2-H-1

SYSTEM DESCRIPTION 1

The CDC CYBER 170 Series, Models 172, 173, 174, and 175 Computer Systems, CDC CYBER 70 Series, Models 71, 72, 73, and 74 Computer Systems, and 6000 Series Computer Systems consist of four logical hardware components. They are:

- Central processor unit
- Central memory
- Peripheral processor units
- Associated peripheral equipment

These hardware elements are controlled and coordinated by two basic levels of software, the system software and user programs. This section describes briefly these hardware and software elements and their relationship within the Network Operating System (NOS).

CENTRAL PROCESSOR UNIT

The central processor unit (CPU) performs computational tasks but has no input/output (I/O) capability. It communicates with the external world through central memory. Under NOS, the CPU is used to assemble, compile, and execute user programs and to perform several system functions and utilities.

The CDC CYBER 170 Series, CDC CYBER 70 Series, and 6000 Series Computer Systems provide two types of central processors. However, the programmer need be concerned only with the distinction between the two types when writing COMPASS programs. Certain instructions, if properly arranged, may be executed simultaneously by the CDC CYBER 170/Model 175 CPU, CDC CYBER 70/Model 74 CPU, and the 6600 CPU. For more information about CDC CYBER 170, CDC CYBER 70, and 6000 systems refer to the hardware reference manuals listed in the preface.

CDC CYBER 170 and CDC CYBER 70 series computers are equipped with a central exchange jump/monitor exchange jump (CEJ/MEJ) feature. This feature enables the system to switch control between the system monitor and a user program. CEJ/MEJ is an option on 6000 series computers. It should be used when available to improve job performance (refer to section 2 of volume 2).

CENTRAL MEMORY

Under NOS, central memory (CM) is used for three basic purposes.

- To hold instructions to be executed by the CPU
- To hold data to be manipulated by the CPU
- To buffer data to and from peripheral processors

NOS supports the following standard CM sizes.

6000 and CDC CYBER 70 - 32K, 49K, 65K, 98K, and 131K

CDC CYBER 170 - 32K, 49K, 65K, 98K, 131K, 198K, and 262K

Several programs can reside in CM simultaneously in hardware protected areas called control points. The fact that these control point areas are hardware-protected means that a program cannot reference an address outside its field length. NOS supports a maximum of 27_8 control points. Each control point may use no more than 3777008 CM words. The user need be concerned only with the memory assigned to his own control point. The system assigns the CPU to the control points requiring CPU activity. Normally, the assignment of the CPU is switched rapidly between the control points to allow all programs in memory to execute. The exact amount of time allowed for each control point depends on system activity and system parameters. Thus, a job may take more real-time to complete at one time than at another. The user has no control over this switching process.

The user program communicates with the system by placing requests in address 1 (RA+1) of the control point (described in section 2 of volume 2). RA is the reference address that specifies the beginning of the user's control point memory area.

When a user program completes, aborts, or is rolled out, the control point is released and made available to another program.

CENTRAL MEMORY RESIDENT

A portion of CM is reserved for system use. This area is, in effect, a control point with special privileges. This area is called central memory resident (CMR). It contains system tables and directories as well as the CPU portion of the system monitor (CPUMTR).

Figure 1-1-1 illustrates the layout of CM and shows the relationship between CMR and the user control points.

EXTENDED CORE STORAGE

Extended core storage (ECS), a second, slower form of memory, is also available. NOS treats ECS as a mass storage device; it can be used:

- For storing frequently accessed data. Refer to ASSIGN Statement in section 7, and Permanent File Control Statements in section 8.

- As an alternate system device for storing copies of ABS, OVL, and PP type routines. Refer to the SYSEDIT control statement in the installation handbook for further information.

The FORTRAN and COMPASS statements for ECS data storage/retrieval are not supported by NOS.

PERIPHERAL PROCESSOR UNITS

The peripheral processor units (PPUs) are small processors that provide communication paths between the central processor and individual peripheral equipment. NOS supports the 10, 14, 17, and 20 PPU configurations of CDC CYBER 170/Models 173, 174, and

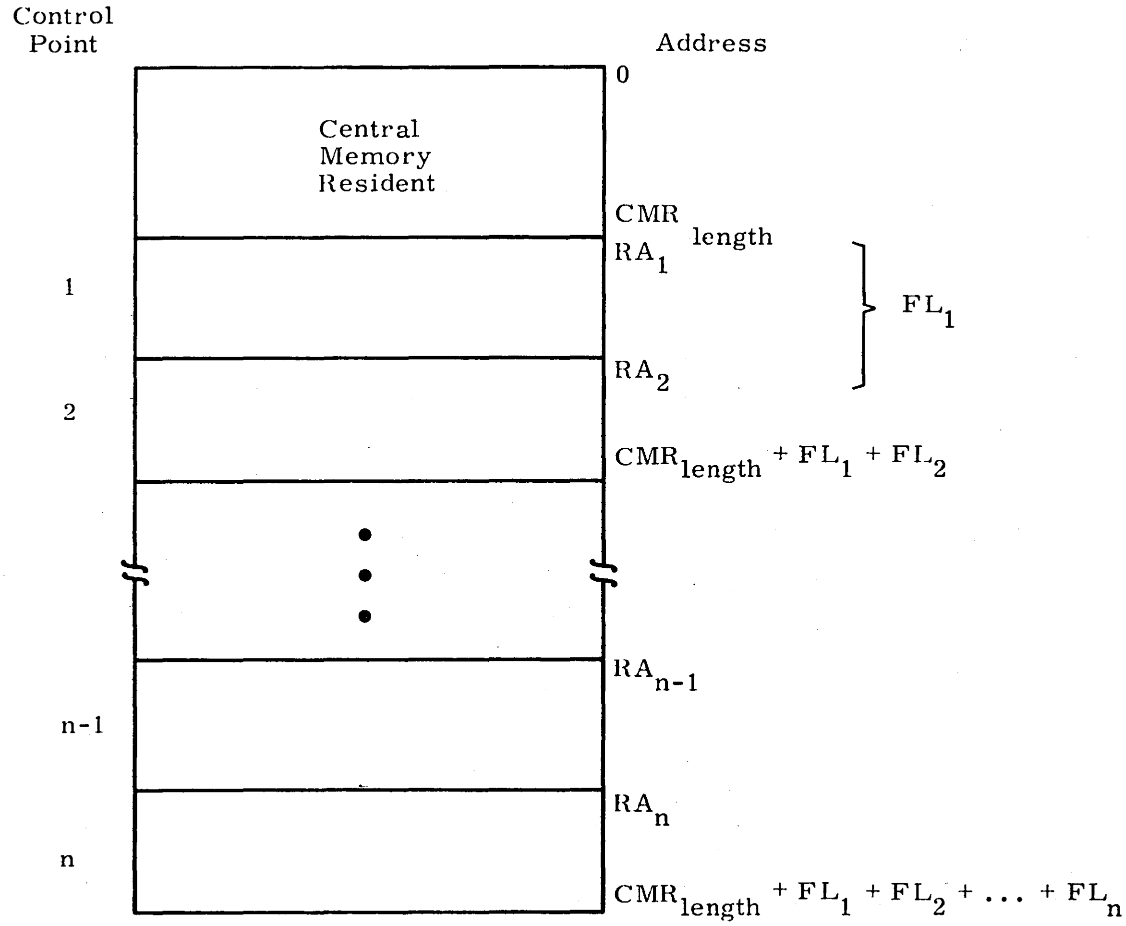


Figure 1-1-1. Central Memory Layout

175 (the Model 172 supports 10 PPUs only) and CDC CYBER 70/Models 71, 72, 73, and 74. The 7, 8, 9, 10, and 20 PPU configurations are supported for 6000 series computers. A peripheral processor can:

- Read and write CM
- Read and write ECS indirectly via CM or directly via the distributive data path (DDP)
- Transmit data to and receive data from peripheral devices using the data channels

The peripheral processors also perform those system control functions that are better handled by a PPU than by the central processor.

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

PERIPHERAL HARDWARE

The system peripheral hardware varies from installation to installation but usually includes card readers and punches, line printers, mass storage devices, and magnetic tape units. The following equipment is supported by NOS.

405 Card Reader

415 Card Punch

512 and 580 Line Printers

841 Multiple Disk Drive

844-21 Disk Storage Subsystem

844-41 Disk Storage Subsystem

844-44 Disk Storage Subsystem

Extended Core Storage

657, 659, 667, and 669 Magnetic Tape Units

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

6671 or 6676 Multiplexers for communication with interactive terminals

2550 Host Communication Processor

The user need be concerned with these devices only to the degree that they affect the format of data being transferred in the system.

SYSTEM SOFTWARE

The system consists of the group of CPU and PPU programs that control the flow of user programs and satisfy any special requests that these programs may make. These special requests include such functions as resource allocation requests and I/O requests.

USER PROGRAMS

A user program is a group of CPU instructions defined by a user to perform a certain task or calculate a specific result. A user program may 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, etc.) 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 scan memory dumps.

FILES 2

A file is the largest collection of information addressable by name. It begins with a beginning-of-information (BOI), an indicator which precedes all data in the file. A file consists of one or more logical records of information. A logical record is a group of related words or characters, of fixed or variable length, which is independent of its physical environment.

The end of a logical record is the end-of-record (EOR). The end of a logical file is the end-of-file (EOF), the end-of-information (EOI), or both. If both, the EOF precedes the EOI. An EOI is the last physical item of information on a file. Because of this EOF/EOI concept, a file may actually be a multifile file. For example:

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

File names are 1 to 7 alphanumeric characters.†

Examples:

A 123 TAPE 1A2B COMPILE

LOGICAL/PHYSICAL FILE STRUCTURE

The actual structure of the BOI, EOR, EOF, and EOI indicators depends on the device on which the information is stored.

The user defines the logical format of a mass storage or magnetic tape file when he issues control statements or language specifications to create the file. Once a file is created, it can be transferred from one storage medium to another without affecting its logical format.

To take advantage of the physical characteristics of the medium on which a file is to be stored, the system converts all user-defined logical file structures into a system-defined physical file structure. In general, for higher-level language users, this conversion process and the resulting physical file format are transparent. All file-related control statements and language specifications transfer data or position a file according to its logical definition. COMPASS users, on the other hand, have the option of reading, writing, or positioning a file according to its logical or physical format.

The basis of all physical file structures is the physical record unit (PRU). The size of a PRU depends on the storage medium used.

† The product set modules (section 11) do not support file names that begin with a numeric.

MASS STORAGE DEVICE FILE STRUCTURE

All data stored on mass storage devices† is written in 64 CM word PRUs. A logical record consists of one or more of these PRUs. The last PRU of a mass storage logical record must be a short (less than 64 CM words) or zero-length PRU.

A BOI for a mass storage file is the disk address for the file listed in the file name table (FNT). An EOR is a PRU containing less than 64 words and having a link to the next PRU in the file. An EOF for a mass storage file is a zero-length PRU (that is, a PRU containing no data) with a special link to the next PRU in the file. An EOI is a zero-length PRU with no forward link. The absence of a link signifies the EOI.

MAGNETIC TAPE FILE STRUCTURE

The operating system uses standard 7- or 9-track, 1/2-inch magnetic tape. BOI on magnetic tape is the load point. The definition of PRUs and of the EOR, EOF, and EOI indicators varies according to the format in which the data was recorded. Any of the following formats can be specified: external (X), blocked (B), line image (E), internal (I), system internal (SI), stranger (S), long block stranger tape (L), and foreign (F), Refer to section 10 for a description of each of these formats.

PUNCH FILE STRUCTURE

Because the physical characteristics of cards define the data, cards do not have a PRU size as previously defined. Refer to appendix F for the conversion procedures used for the various types of punch cards. The logical format of the file is indicated as follows:

- The first card in the deck is the BOI
- A 7/8/9 punch in column 1 represents an EOR
- A 6/7/9 punch in column 1 represents an EOF
- A 6/7/8/9 punch in column 1 represents the EOI

Thus, a deck can consist of many files which can consist of many records, as illustrated in figure 1-2-1.

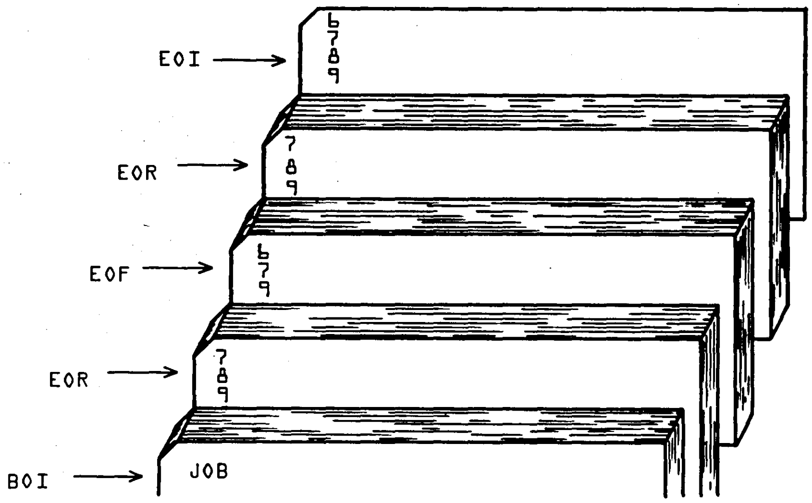


Figure 1-2-1. Sample Card File Structure

† ECS files are allocated in the same manner as all mass storage files.

FILE TYPES

Active files in the system are classified by their file type. Whenever a file is active, one or more entries are made in the file name table (FNT). The FNT entry and the file status table (FST) entry comprise a two-word description of the file. These two entries contain the name of the file, the device on which the file resides, the file type, the current position, and the current status. All system tasks involving a file use this two-word entry for control.

In each of the following descriptions, the file type and its mnemonic (such as INFT) which the system uses internally for file classification are listed.

QUEUE FILES

Five types of files are defined as queue files. They are categorized as such because of the kinds of information they contain and the manner in which the system processes them. Queue files always reside on mass storage. When a queue file is ready to be processed, the system or the user places it in a queue where it waits until the required system resource or peripheral equipment becomes available.

Input Files (INFT)

Input files are the job files of the system. They contain all user-supplied control statements and program data. There are two ways a file can be placed in the input queue, directly by the system in initiating a local or remote batch job for processing and indirectly by a user job in submitting another job via a SUBMIT control statement, an LDI control statement, or a ROUTE control statement.

When central memory space becomes available, either because a job has completed or because a job in the input queue has a higher priority than that of a job being processed, the input file is scheduled for processing (in other words, the job is assigned to a control point in central memory). Refer to section 3 for a description of the elements of jobs and the processes of job initiation and scheduling.

Rollout Files (ROFT)

At some stage in the processing of a job, the system or the user may determine that the job must be temporarily removed from central memory. When this occurs, the system writes all information concerning the job on a system-defined rollout file. The rollout file includes the contents of the job's central memory field length and the job-related system information from CMR. The file is read back into central memory when the job is again scheduled at a control point. (Refer to Rollout Control in section 3.)

Timed/Event Rollout Files (TEFT)

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

A job may be rolled out on a TEFT file as a result of system or user action. The system uses a timed/event file if a job issues certain requests for a file or device that cannot be immediately honored. The COMPASS programmer can use the ROLLOUT macro to roll out his job subject to specified time and/or event dependencies.

Print Files (PRFT)

A print file contains data the user wishes to have printed during his job or upon job completion. The system-assigned name for print files is OUTPUT.† OUTPUT is placed in the print queue either by the system when the job completes or by the user via an OUT control statement. The user can also utilize a ROUTE or DISPOSE control statement.

Once a file enters the print queue, it is processed by the local or remote batch printer processor. Then, when a printer becomes available, the PRFT file with the highest priority is printed.

Most system utility reports are written on OUTPUT unless the user specifies an alternate file. OUTPUT has no special internal format. Refer to appendix F for a description of conversion methods and printer control characters and to appendix D for a description of job output information.

Punch Files (PHFT)

Punch files contain data that the user wishes to have punched on cards during his job or upon job completion. The system-assigned names for punch files are:

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

These files are released to the punch queue when the job completes. In addition, the user can utilize an OUT, ROUTE, or DISPOSE control statement in the same manner as described for PRFT files to place a file in the punch queue.

Refer to appendix F for a description of the format of the PUNCH, PUNCHB, and P8 files.

SPECIAL FILES

Of the five special files, the first two described (local and direct access permanent files) are general-purpose, and the remaining three (library, system, and primary terminal) are special-purpose.

Local Files (LOFT)

All scratch and working files are designated as local files. The user can create a local file in three ways; he can:

1. Implicitly create a local file by making the first reference to it in one of the COPY control statements, any read or write language specification, or an OPEN

† For time-sharing jobs, the name OUTPUT has special meaning. Refer to section 12, volume 2, and to the Time-Sharing User's Reference Manual.

macro. Local files created in this manner always reside on mass storage.

2. Create a local file by preceding any COPY statements, read or write specifications, or OPEN macros with an explicit control statement or macro file definition. The ASSIGN control statement or the REQUEST control statement or macro assigns a local file to mass storage or magnetic tape. The LABEL control statement or macro assigns a local file to magnetic tape.

3. Use a GET control statement or macro to generate a local mass storage copy of an existing indirect access permanent file. For a description of indirect access permanent files, refer to Permanent Files in this section.

Unless the user includes a control statement or macro to change a local file to another type of file, it is released upon job completion.

Direct Access Permanent Files (PMFT)

A direct access permanent file is the type of permanent file that can be accessed directly rather than through the use of a working copy. The user creates a direct access file with the DEFINE control statement or macro. Once the file is created, the originator or anyone else to whom the originator has given permission can assign the file to his job with an ATTACH control statement or macro. The file remains in the system until the originator removes the file with a PURGALL control statement, or the originator or any other user with the necessary permission removes the file with a PURGE control statement or macro.

For further information about direct access permanent files and their relationship to indirect access permanent files, refer to Permanent Files in this section.

Library Files (LIFT)

A library file is a read-only file that can be accessed by several users. A user must be validated to access/create library files. Note that this type of file should not be confused with system library programs or public permanent files (user number LIBRARY).

A library file is created by performing the following steps.

1. Create a local file lfn.
2. Enter the following directives as control statements or macros.

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

If a user wishes to read this file and knows the file name, either the COMMON control statement or ASSIGN macro is entered. When either of these functions is performed, an FNT entry representing this file as a library type file is created.

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.

For a description of the relationship between LIFT files and other libraries and library files, refer to Libraries in this section.

System Files (SYFT)

The system uses SYFT files for retaining special system information. SYFT files always reside on mass storage. Although the COMPASS programmer who is validated to create system files can do so with an ESYF macro, only special system programs can access them. Once a system file is created, no user including the originator can remove it. However, system files are lost at system deadstart unless the operator recovers them.

Primary Terminal Files (PTFT)

The primary file is the main working file for the user. Of several files which may be local to his job, the user may designate one file to be the primary file by using a NEW or PRIMARY statement. (A copy of an indirect access file may be retrieved and made a primary file using the OLD statement.) This becomes the default file if a file name is not specified. Only one primary file is available to the user at a time.

QUEUED FILE MANAGEMENT

Queued file management routes all output generated by a job to the remote batch terminal where the job was created. This is implemented by assigning a unique terminal identification code (TID) every time a remote batch terminal is logged in. (The TID can be the user index associated with the user number when logging in.)

When a job from a remote batch terminal is put into the input queue, the TID is associated with the job input file. This TID is transferred to any print or punch file that is generated by job processing. At job termination, the system uses the TID to route the output file to the remote batch terminal of job origin. In a similar way, a central site card reader may be assigned an identification number (ID) to route batch output files to a line printer or card punch with the same ID.

The user can route output files to a particular device rather than accept the TID default. The particular device can be specified on the basis of device type (printer model, punch format, etc.), external characteristics (ASCII, 029, etc.), or forms. code (requires special form selection). Refer to ROUTE control statement, section 7 for a description of file routing.

RESERVED FILE NAMES

Several file names are reserved for system use or have special significance to the system. These reserved names protect the user from accidentally destroying some of his own files. The message

RESERVED FILE NAME.

indicates that the user has attempted to use one of these files.

The reserved file names are:

INPUT
OUTPUT
PUNCH
PUNCHB
P8
SCR
SCR1
SCR2
SCR3
SCR4

A number of the product set members, such as the COMPASS assembler, use other scratch files in addition to those listed.

Since system integrity does not depend on preventing the use of these reserved file names, an attempt is not always made to prevent their use.

PERMANENT FILES

The user can create, retain, and access files which are available until he specifically decides to remove them from the system. These files are called permanent files. There are two types of permanent files.

- Direct access permanent files are accessed using normal I/O procedures, including random read and write requests. Direct access permanent files are allocated in large blocks;† thus, they are generally used as large data base files. Direct access files have a write interlock. This means that if one user has attached the file in write mode, it cannot be attached by another user. Likewise, if a user wishes to attach the file in write mode, he must wait until all current users have completed using the file. The user should also note that because data is written directly on the file rather than on a working file, care must be taken when modifying a direct access file.

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

- Indirect access permanent files are accessed by using a working copy of the file as a local file attached to the user's job. This working copy is obtained with the OLD control statement or the GET control statement or macro. If the user wishes the working copy to remain permanent after the file has been altered, the SAVE or REPLACE functions must be issued. Indirect access files are allocated in blocks of 64 central memory words (640 characters). Because of this smaller block size and the convenience of a working copy, the indirect access file is generally the method used to create a small permanent file that does not require a write interlock.

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

User access of permanent files is based on the user number entered with the USER control statement. This user number is a 1- to 7-character value which represents a specific catalog in the permanent file system. Unless otherwise specified by an optional (alternate) user number, all permanent file requests are made to this catalog.

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

UABCDEF

UABDDEE

MABCDE1

MAB1DE3

† Refer to Permanent File Device Statistics, appendix E.

DEVICE RESIDENCE

For most 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 system 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 magnetic tape or mass storage device or to one of a type of magnetic tape or mass storage devices.

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.

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 indirect access files, and 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 in multimainframe mode. 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 of specifying another type of permanent file device.

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.

ACCESSING FILES

The two methods used to access files attached to a job are sequential and random access. Any file can be accessed sequentially; however, only mass storage files can be accessed randomly.†

To read a file randomly, the system reads a portion of the file without reading all information in the file, from the current position to the desired position. Any mass storage file can be read randomly if the user knows which relative PRU (that is, which PRU in relation to the BOI) he wishes to read. The desired PRU can be read by placing the PRU number in the file's communication area (FET) and making the proper I/O requests (refer to section 3, volume 2).

Several methods of random processing exist. The specific method depends on the language being used; however, in all cases, the following points apply.

- Most random I/O operations require a directory or index that contains the relative PRUs of records in the file.
- An EOR or EOF I/O operation transfers one PRU for the EOR or EOF.
- When randomly rewriting data within a file, the user must take care to ensure that data following the area he wishes to write is not destroyed.

Figure 1-2-2 illustrates a typical example of the structure of a random access file.

READING FILES

To read record 3 sequentially, the program rewinds the file to BOI, reads the file, and counts the number of EORs. System utilities and macros can be used to skip the records; however, the primary consideration is that the data must be read to determine where record 3 begins. Once this is determined, record 3 can be read.

If a directory exists for this file, the only requirement is that the random address of record 3 be obtained from the directory and placed in the FET. The proper random read requests can then be issued. To perform this random read on record 3, the following steps are required.

- Skip to the EOI. This is done by the system without reading the entire file.
- Backspace two logical records (one record for the EOF and one for the directory). The system must read both records to perform this operation.
- Read the directory to obtain the random address to be placed in the FET.

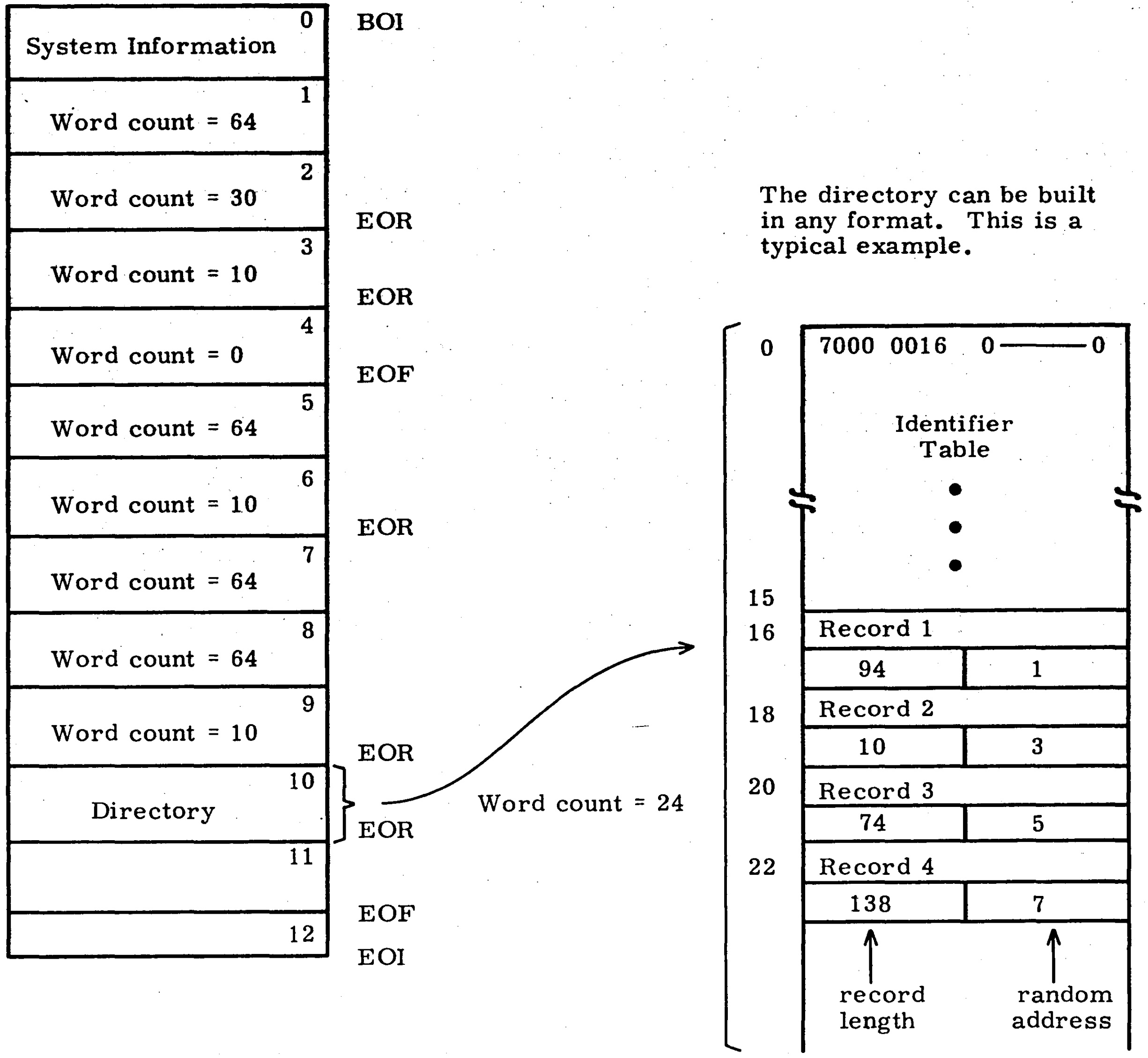
NOTE

> The EOF may or may not be used at the end of this file. The language and methods used to build the directory determine whether an EOF is used.

In summary, to access record 3 sequentially, four PRUs must be read. To access the record randomly, only three PRUs are read: two PRUs to position for the directory and one PRU to read the directory.

For additional random accesses to any record in the file, it is not necessary to access the directory again if it remains in central memory.

† Record Manager random files are treated sequentially by the operating system.



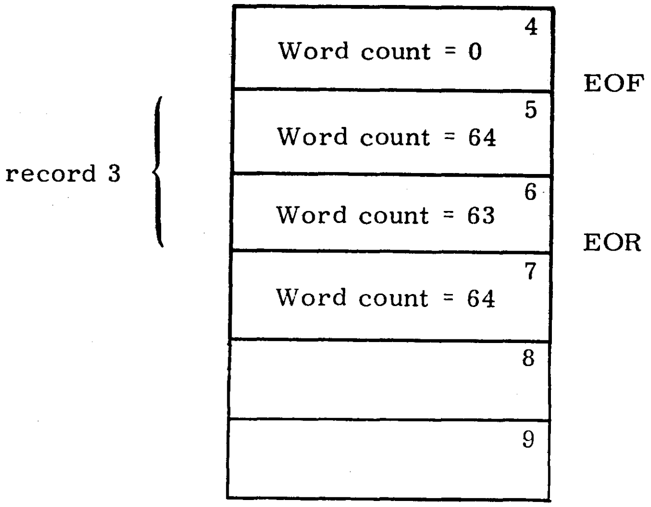
The directory can be built in any format. This is a typical example.

Figure 1-2-2. Sample Random Access File Format

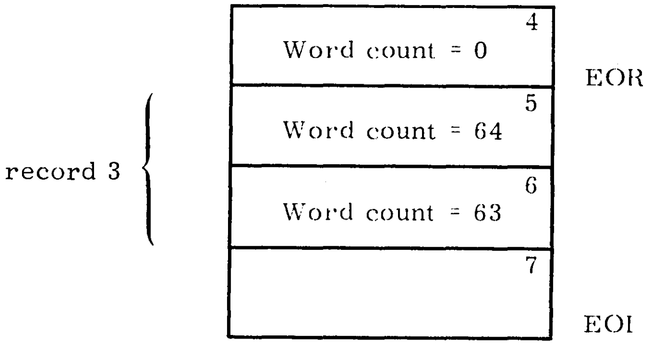
Each directory entry contains the record name, the first PRU of the record (random address), and the record length. The directory can be placed anywhere in the file. The only requirement is that those users who wish to access the file randomly know where to position the file in order to read the directory. However, the directory usually precedes the EOF/EOI.

WRITING FILES

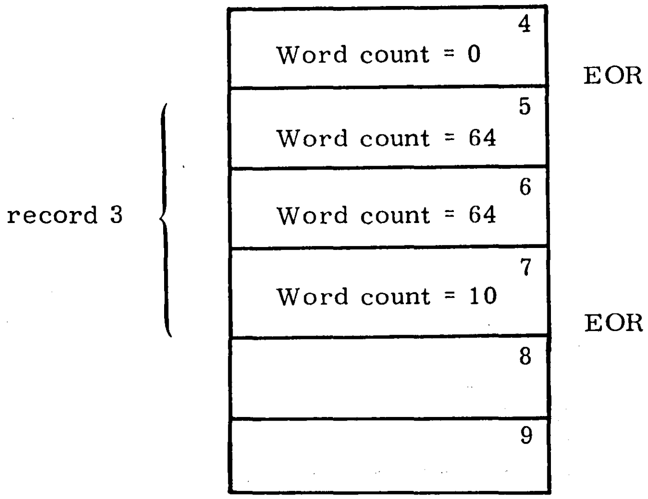
After reading and modifying record 3 of the sample file, the user may wish to rewrite the record in the file. If the modifications have not changed the number of PRUs required, a write operation can be used to replace the existing record with the modified record. This write operation must be issued as a random I/O operation. (Refer to section 3, volume 2 for a complete description of the method.) However, if the modifications have changed the number of PRUs required, data following the record being written is lost. For example, the size of record 3 in the sample file is 74 words or two PRUs. A maximum of 53 words can be added to the record without requiring an additional PRU and destroying data. If a random write request that adds 53 words to record 3 is issued, the file has the following format.



This operation is called a rewrite in place. If the write is issued as a nonrandom write operation, the file has the following format.



All data following the inserted data is destroyed. If the word count for record 3 is increased to 138, the file has the following format.



PRU 7 is destroyed by the write operation. To properly rewrite record 3 without destroying the contents of PRU 7, the user should issue a write request at the end of the file and alter the directory to reflect the change. Figure 1-2-3 illustrates the updated file containing the new directory and the 138-word modified record 3 written at·the end of the file.

LIBRARIES

The term library can be used in four ways in the system. The following paragraphs define the various types of libraries and the methods, if any, by which the user accesses them.

- System library. The system library consists of the assembled routines that comprise the operating system and its associated product set. System routines may reside in central memory, mass storage, or ECS. The user accesses the system library indirectly when a system routine is executed in response to a control statement or macro call. A complete copy of the system library is saved on a read-only file named SYSTEM. Refer to the CATALOG control statement in section 7 for a partial list of the system library routines.
- Program library. A program library is a group of source deck images saved on a program library file in compressed format. There are two system-defined program libraries, OPL and OLDPL. OPL contains operating system routines saved and maintained in Modify format via the MODIFY control statement. OLDPL contains product set routines saved and maintained in Update format via the UPDATE control statement. In addition, the programmer can use a MODIFY or UPDATE control statement to create and edit his own program library.
- User library. Before a user's compiled program can be executed, all external references must be satisfied. The loader satisfies externals by searching user libraries. A user library is a group of compiled or assembled object time routines saved on a user library file. There are three types of user library files: user-generated, product set, and system.

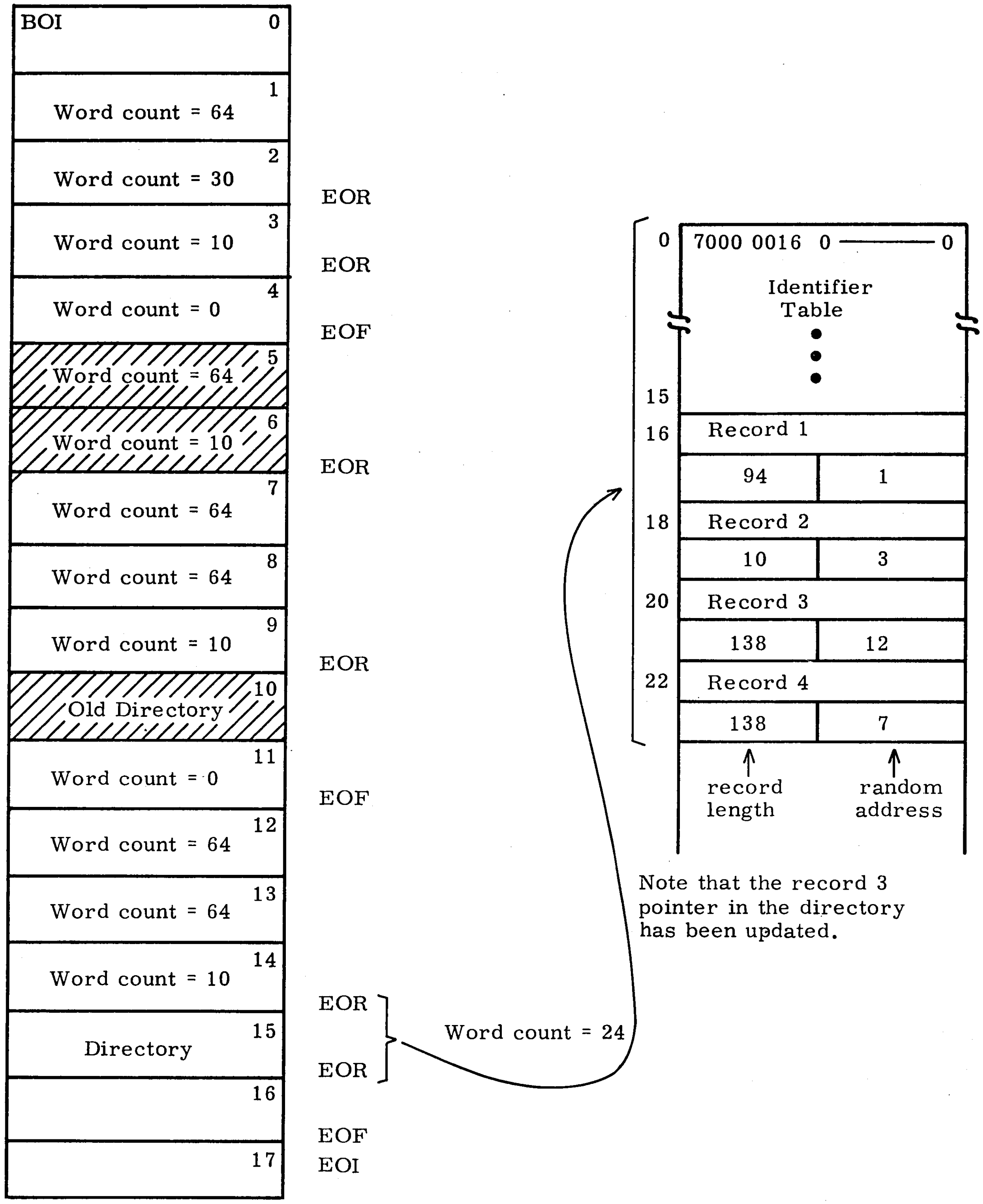


Figure 1-2-3. Modified Sample Random Access File

Appendix B, volume 2 contains examples of COMPASS programs that create, read, and write a random file.

User-generated libraries are created with the LIBGEN control statement and can be specified on LIBRARY and/or LDSET control statements. Refer to the Loader Reference Manual for further information. Product set libraries reside as ULIB type records on the system library. They are listed in section 11. If some externals remain unsatisfied after searching these libraries, the loader searches the system default user library SYSLIB, which also resides as a ULIB record on the system library.

- User number LIBRARY. An installation can save under the user number LIBRARY permanent mass storage files containing programs or text of general interest (such as application programs and procedure files).

JOB FLOW AND EXECUTION 3

A job consists of a file of statement images grouped into several records. The first logical record contains the control statements that specify the job processing requirements. Each control statement is an individual job step. Every job begins with a job statement and ends with an EOI. All other control statements directly follow the job statement. The end of the control statements is marked by an EOR, EOF, or an EOI. Figure 1-3-1 illustrates a basic job deck.

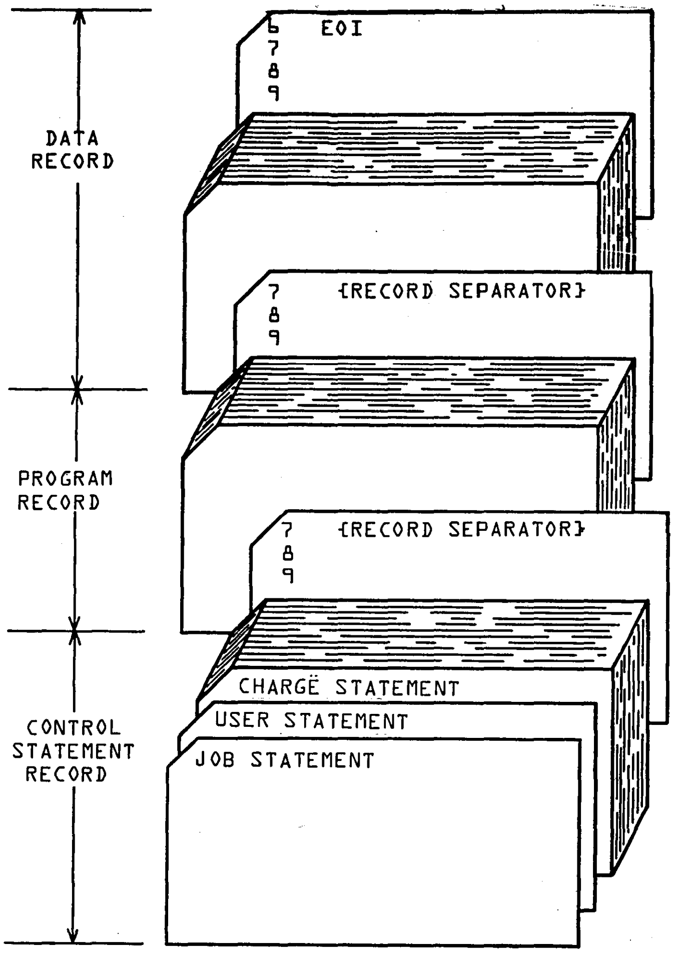


Figure 1-3-1. Basic Job Deck

Figure 1-3-2 illustrates a COMPASS source deck that produces the object code and a listing and executes the binary file using the input data supplied.

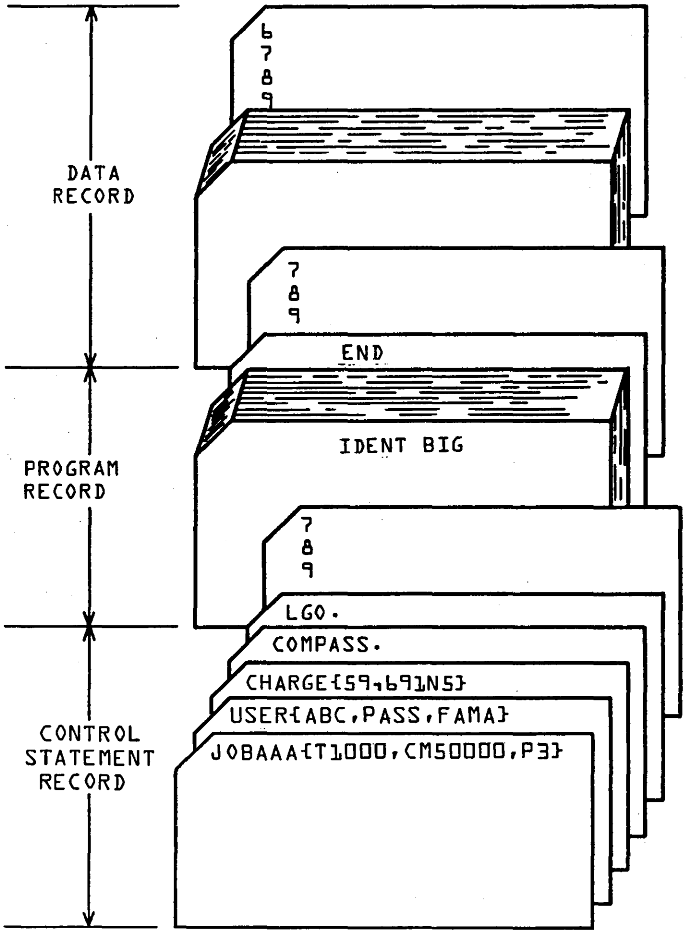


Figure 1-3-2. COMPASS Assemble and Execute Deck

Figure 1-3-3 illustrates a COMPASS deck that assembles the program, produces binary punched files of each subprogram, and executes the object code of the first program record.

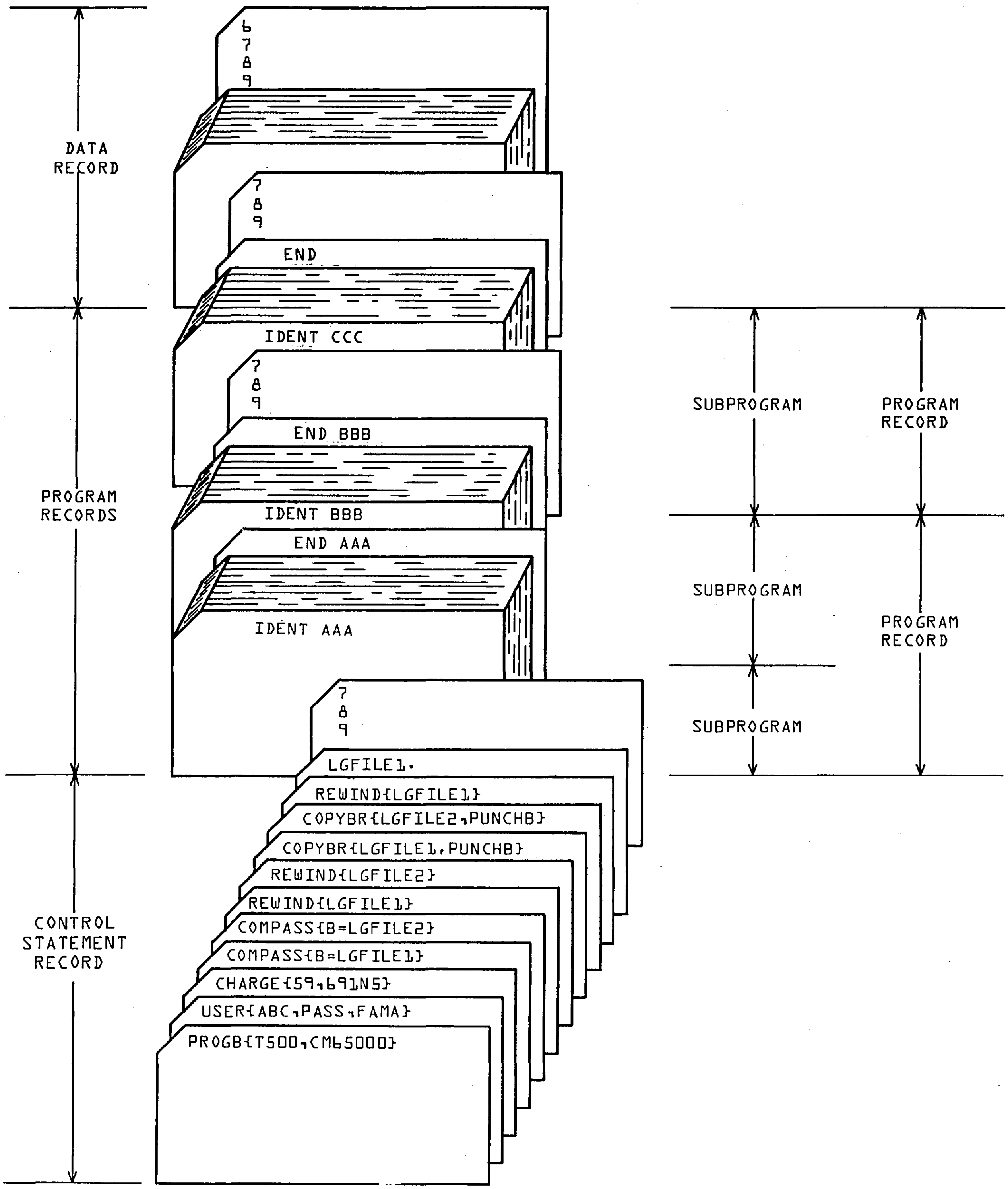


Figure 1-3-3. COMPASS Assemble, Execute, and Punch Binary Deck

Figures 1-3-4 and 1-3-5 illustrate examples of FORTRAN source decks used for computation and user output.

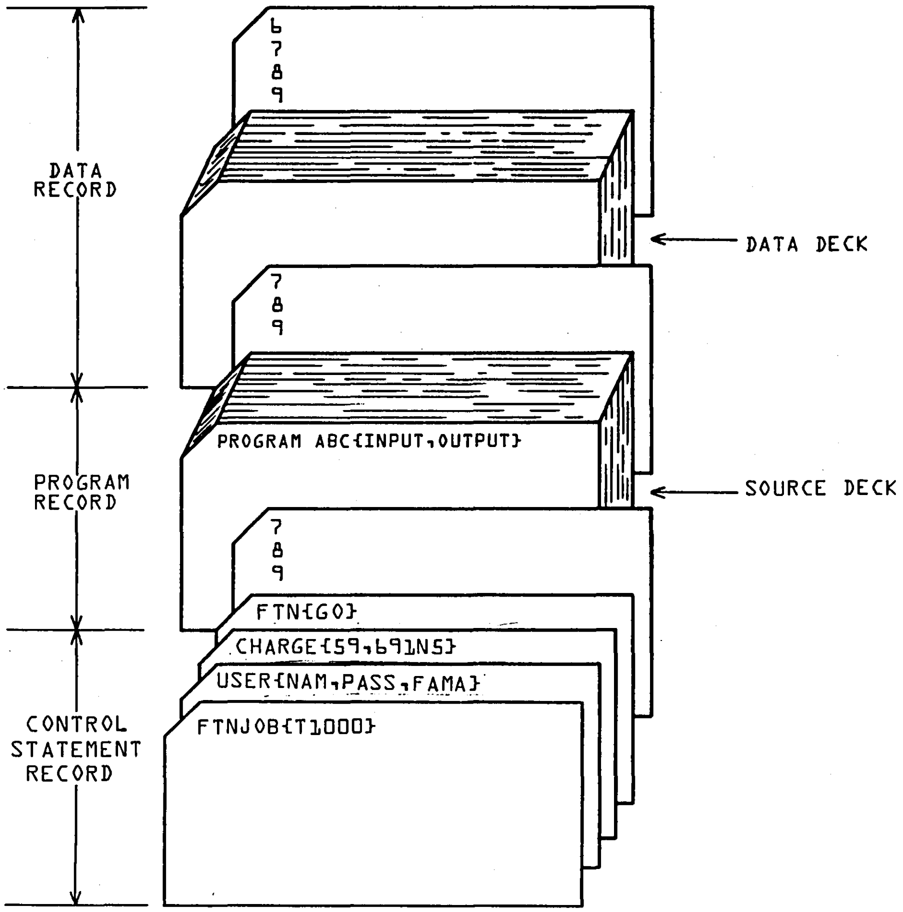


Figure 1-3-4. FORTRAN Compile and Execute Deck

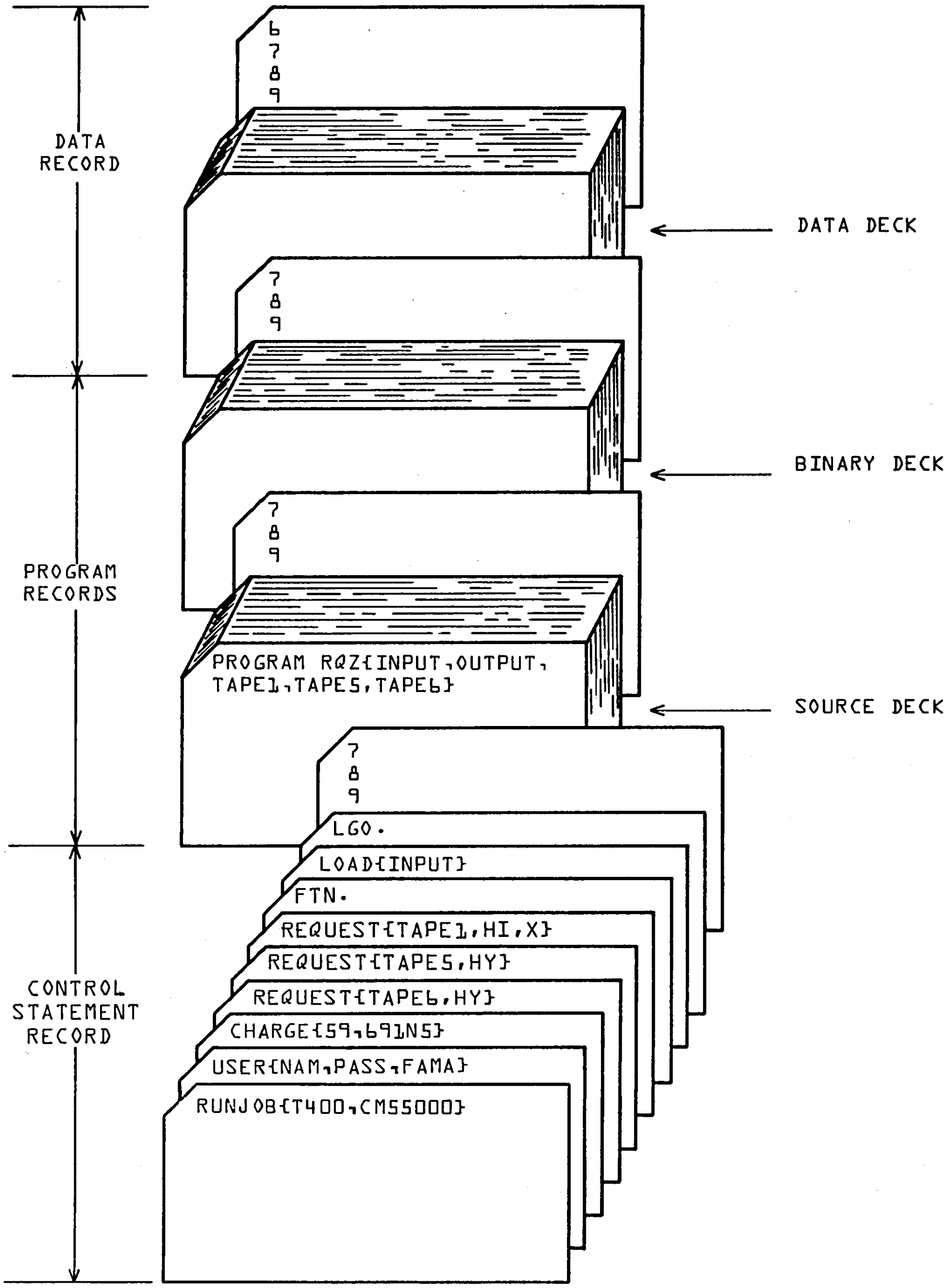


Figure 1-3-5. FORTRAN Load and Run Deck

JOB INITIATION

When a job enters the system, the system determines the job origin type. The job origin type identifies the means by which the job is entered into the system. It is also used to identify the job while it remains in the system. The job origin type is used by the system to control job activity and to aid in directing the job through the system. It also determines the way the job exits from the system.

Jobs are initiated by:

- Reading a card deck in through a card reader, either from a local or a remote batch reader.
- Logging into a time-sharing terminal.
- Using the available load utilities (refer to the LDI control statement, section 6).
- Using the SUBMIT control statement from a job already in the system (refer to the SUBMIT control statement, section 6).
- Using the ROUTE control statement from a job already in the system (refer to the ROUTE control statement, section 7).

The first two methods of initiation set the job origin type to indicate the method by which the jobs are entered. When using the SUBMIT or ROUTE control statement, a parameter is entered with the command specifying the origin type.

JOB ORIGIN TYPES

If the job originates from the system console, the job is assigned system origin type (SYOT). If the job is a time-sharing job and enters through the time-sharing executive, it is given time-sharing origin type (TXOT). If the job enters the system through the local card reader, it is a batch origin type (BCOT) job. A job coming into the system from remote batch is entered into the system by the Export/Import package or the Remote Batch Facility (RBF) package and is assigned Export/Import origin type (EIOT).

If validated, a user can submit jobs to the system using the SUBMIT or ROUTE control statements. The user can also specify origin types that are different from his own.

JOB NAMES

After entering the system, the job is assigned a unique job name to prevent job name duplication within the system. This job name is a combination of parameters that describe the job; the first seven characters 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 four characters of a system job name are obtained from the job name entered or are zero-filled if fewer than four characters are entered. The next three 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 four 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 four characters are unique to the user. The next three characters are the job sequence number. The eighth character of a batch origin job name is B.

TIME-SHARING AND EXPORT/IMPORT (TXOT AND EIOT) JOB NAME FORMAT

The first four 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 three 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 four characters of their job names from the job's current user index in the same manner as EIOT and TXOT jobs. The last three digits 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, 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 on which permanent file device (and where) the user's permanent files reside. (Refer to part IV of the installation handbook 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 his 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.

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 whenever 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 of 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

Before the system begins to process a job step (control statement), it establishes the initial field length for that step from the first one of the following that applies.

1. The field length requirement is specified by RFL= and MFL= special entry points. This is true of several system routines (refer to appendix F, volume 2).

2. The routine contains the field length required in a loader table (54 table).

3. An initial running field length (RFL) has been specified with an RFL control statement or a SETRFL macro call.

4. Either the maximum field length (MFL) for the job step or the system default (50000B) is chosen, depending on which is the smaller.

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

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

- The field length specified on the job statement
- The maximum field length for which the user is validated
- The maximum field length available to the user for processing of this particular job (machine size)

For each job step, there is an MFL, which cannot be exceeded during processing of that step. MFL may be reset between job steps with the MFL control statement; however, MFL cannot exceed MAXFL, the maximum field length for the job.

Assuming no other overriding information (RFL= or MFL= entry points or 54 loader table), the system begins a job step with the initial running field length (RFL) as set by a previous RFL control statement or SETRFL macro call. RFL may not exceed MFL, the upper bound for the particular job step. After processing of the job step has begun, additional field length may be acquired with the MEMORY macro call; however, the field length cannot be increased above MFL.

The levels of maximum field length may be summarized with the formula

$RFL \leq MFL \leq MAXFL$

which states that the initial running field length cannot exceed the maximum field length for a job step which cannot exceed the maximum field length for a job.

The flow of processing to determine field length for each job step is shown in figure 1-3-6.

Examples of field length control:

Control Statement	Field Length	Comments
JOB(CM60000, T1000)	700	MAXFL and MFL are both set to 60000. RFL is set to 0, indicating system control of FL. Field length is set to 700 to allow ACCFAM to run.
USER(USERABC, 1234567, FAM1)	700	ACCFAM, which processes the USER control statement, has an RFL= special entry point.
RFL(50000)	400	The RFL control statement sets RFL at 50000, indicating an initial running field length for job steps with no other overriding considerations. The field length is set to 400, because the utility that sets the running field length (CONTROL) has an RFL= special entry point.
ATTACH(OPL=OPLX)	1600	Field length is set to 1600, because the utility that attaches files (PFILES) has an RFL= special entry point.

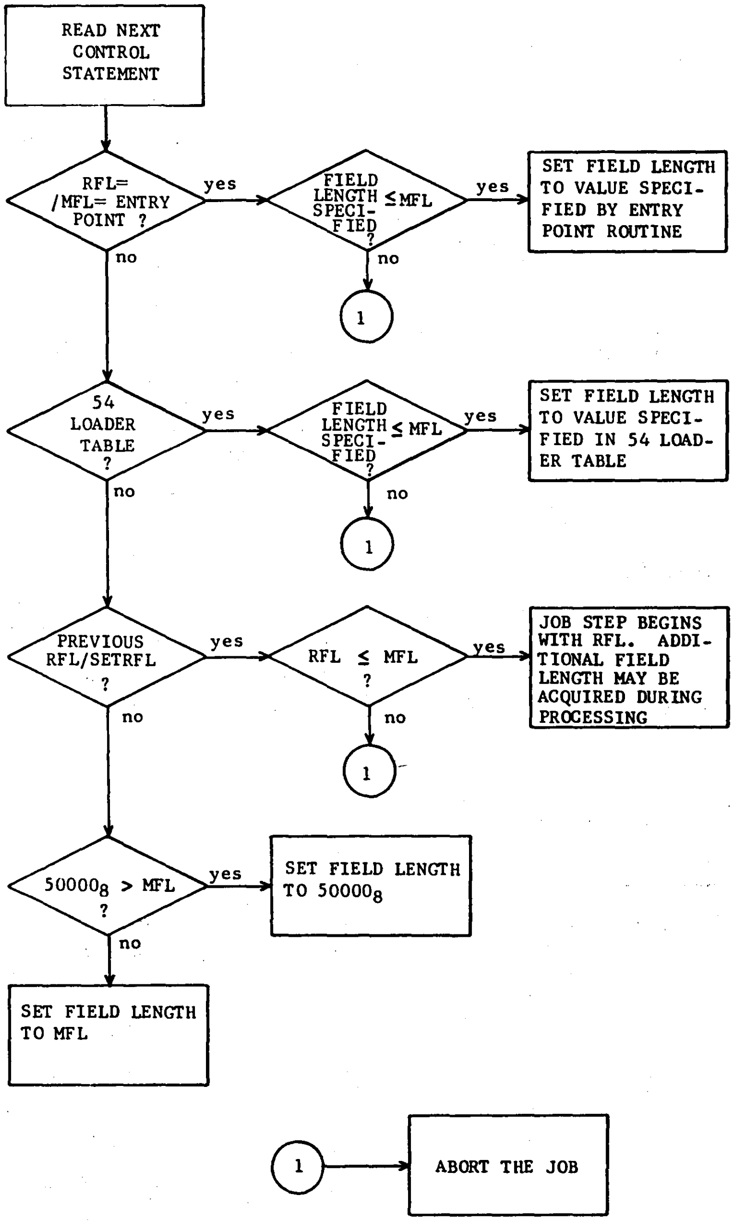

READ NEXT CONTROL STATEMENT
RFL= /MFL= ENTRY POINT ?
yes
FIELD LENGTH SPECIFIED ≤ MFL ?
yes
SET FIELD LENGTH TO VALUE SPECIFIED BY ENTRY POINT ROUTINE
no
no
1
54 LOADER TABLE ?
yes
FIELD LENGTH SPECIFIED ≤ MFL ?
yes
SET FIELD LENGTH TO VALUE SPECIFIED IN 54 LOADER TABLE
no
no
1
PREVIOUS RFL/SETRFL ?
yes
RFL ≤ MFL ?
yes
JOB STEP BEGINS WITH RFL. ADDITIONAL FIELD LENGTH MAY BE ACQUIRED DURING PROCESSING
no
no
1
50000_8 > MFL ?
yes
SET FIELD LENGTH TO 50000_8
no
SET FIELD LENGTH TO MFL
1
ABORT THE JOB

Figure 1-3-6. Field Length Control Processing

Control Statement	Field Length	Comments
MODIFY(L=0, Z)/*EDIT, FORT	36600	Modify increments in 2000-word blocks to the table size required to complete the Modify operation.
FTN(I)	50000	FTN has a pseudo MFL= entry point supplied through the FL directive of SYSEDIT. This has a value of 42000. The 50000 from the last RFL statement is the larger value and is used.
LGO.	15000	The loader automatically reduces the job field length after performing a relocatable load. (RFL= is not present in LGO.) The loader requires a field length of 30200 to load itself. It increments in 4000 word blocks until the required table space is available.
FTN(I=TAPE1)	50000	FTN has a pseudo MFL= entry point supplied through the FL directive of SYSEDIT. This has a value of 42000. The 50000 from the last RFL statement is the larger value and is used.
REWIND(TAPE1, COMPILE)	1000	The system sets the field length because REWIND utility has RFL= set. (FILES is the utility package.)
SAVE(LGO=BIN)	1600	The SAVE utility also has RFL= set. (PFILES is the utility package.)
MFL(40000)	400	MFL is reset to 40000. No following job step can exceed this value of MFL. The MFL control statement is also processed by CONTROL. RFL is cleared.
RFL(30000)	400	RFL is set to 30000.
LIBEDIT(P=0, N, V)	30000	Because LIBEDIT has no RFL= specified, the system restores the field length to the running field before processing LIBEDIT.

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 attached 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 specifies a job step time limit. This time is the amount of central processor time that any one job step is allowed. The maximum time allowable on the job statement is 77770_8 seconds. Any job in the system with a time limit of 77771_8 through 77777_8 seconds has an infinite amount of central processing time at its disposal. If the user wants to change a job's time limit, the SETTL control statement or macro is used. The user cannot, however, 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 Time-Sharing User's Reference Manual for a more detailed description.

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 job 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 Time-Sharing User's Reference Manual for more details.)

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

The exit mode feature allows the programmer to select conditions that permit the system to discontinue normal processing when errors occur. The error conditions and associated condition codes that can occur are:

- Illegal instruction (00) — The CPU attempted to execute an illegal or nonavailable instruction.
- Address is out of range (01) — One of the following conditions has occurred.

  The program attempted to reference CM memory or ECS outside the established limits.

  The program is attempting to branch to an address outside the user's field length.
- Operand is out of range (02) — Floating-point arithmetic unit received an infinite operand.
- Indefinite operand (04) — Floating-point arithmetic unit attempted to use an indefinite operand.
- ECS flag register operation parity error (10)† — Parity error was detected on ECS flag register operation.

†Applicable to CDC CYBER 170 series only.

- CMC input error (20)† — The address or data sent by the CPU had incorrect parity at the central memory control (CMC) or CM.

- CM data error (40)† — Double data error (two data bits failed) between the CMC and CM, detected by the single-error correction double-error detection (SECDED) network, or a data parity error between the CMC and CM when operating in default mode (in other words, the SECDED network has been disabled).

The user can select any combination of these conditions with the MODE control statement (refer to section 6). If one of these errors occurs and the proper mode for that error is selected, the system notes the error by setting the appropriate error flag and exiting from normal processing. The following dayfile error message occurs defining the error exit conditions.

```
CPU ERROR EXIT xx AT yyyyyy.
```

This message identifies the error condition by the condition code xx (as listed above) that was detected at location yyyyyy. If the exit mode is not selected, the central processor stops or proceeds depending on the situation. For a detailed explanation, refer to the appropriate hardware reference manuals.

When activity at a control point ceases, the system determines the reason. If an error flag is set, the error is noted and execution is resumed at the error exit address if one was specified. Error exit addresses are set by the EREXIT macro (refer to section 6, volume 2).

Once control is transferred, the error flag is cleared. If the error occurs because the central processor time limit is exceeded, the job is given another 10_8 seconds to complete processing. If the error is caused by a central processor abort (refer to ABORT macro), the address at which the error occurred is specified and normal error processing continues.

When control is transferred from an executing program because of an error, the system determines whether or not to continue with control statement processing, perform error processing, or terminate the job.

The system first searches for an EXIT control statement. If an EXIT statement is found, error processing begins with the statement following EXIT. If, prior to the detection of the error, the system encountered a NOEXIT statement, no search is made for an EXIT statement and processing continues with the next control statement. If no EXIT or NOEXIT statement was encountered, the system terminates the job.

† Applicable to CDC CYBER 170 series only.

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 2, 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).

Violation of these restrictions results in the control statement being ignored and the following informative message being entered in the dayfile.

```
SECURE MEMORY, DUMP DISABLED.
```

The following are the restricted control statements.

CATALOG
CKP
COPYB
COPYC
DMD
DMP
EDIT
LBC
LIBEDIT
LOC
PBC
RBR
VERIFY
VFYLIB
WBR
RESTART

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. All equipment assigned to the job is returned to the system.

3. All library files attached to the job are returned ; other jobs can then access them.

4. All scratch (local) file space used by the job is released.

5. All direct access permanent files attached to the job are returned; the status information for these files is updated.

6. 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, es, xxxxxx.xxxKLNS.

es	EST ordinal of the file

   The following information is issued to the account dayfile only:

   - Cards read in kilocards:

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

   - Cards punched in kilocards:

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

7. Control point dayfile is copied to the end of the print file.

8. All output files are released to the output queue.

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

CONTROL LANGUAGE 4

The operating system control language allows the programmer to transfer control and to perform arithmetic and test functions within the control statement record. Control language consists of statements similar to FORTRAN statements. These statements are normally composed of a command (as listed below), parameters, symbolic names, and expressions. The following are legal commands.

GOTO
CALL
DISPLAY
SET
IF

An important feature of control language is the capability to create procedure files. A procedure file is a group of system control statements and/or control language statements which can be called much like a subroutine for insertion anywhere within the control statement record. It is activated either by the CALL statement or the name of the procedure file. Because control statements, control language statements, or both are allowed in a procedure file, the user is given a much wider range of control for manipulating his files.

The following sections describe the various components and commands of the system control language.

EXPRESSIONS

The expressions allowed are similar to FORTRAN expressions and may contain constants, arithmetic operators, relational operators, Boolean operators, functions, and symbolic names.

CONSTANTS

Numeric constants are assumed to be decimal. If a constant has a postradix of D, it is decimal. If it has a postradix of B, it is octal.

ARITHMETIC OPERATORS

Arithmetic operations are performed in ones complement with 48-bit evaluations. The arithmetic operators processed are:

+	Addition
-	Subtraction
*	Multiplication
/	Division
**, ↑	Exponentiation

Leading -	Negation
Leading +	Ignored

RELATIONAL OPERATORS

Relational operations produce the value 1 if the relation is true and a value of 0 if the relation is false. The relational operators are (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

BOOLEAN OPERATORS

The Boolean operators are (either form may be used):

≡	.EQV.	Equivalence
∨	.OR.	Inclusive OR
∧	.AND.	AND
↓	.EOR.	Exclusive OR
¬	.NOT.	Complement

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

CPE	CPU abort
EIO	Remote batch (Export/Import) origin
FLE	File limit error
MNE	Monitor call error
ODE	Operator drop
PEE	CPU parity error exit
PPE	PPU abort
PSE	Program stop error
SRE	SRU limit error
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; in expressions, SS may be equivalenced to one of the following. ACCESS† BASIC BATCH EXECUTE FTNTS NULL TRANACT†

- Symbolic names with Boolean values:

F	False value
FALSE	False value
SWn	Setting (1=on, 0=off) to sense switch $(1 \leq n \leq 6)$
T	True value
TRUE	True value

†Special validation is necessary to access and use ACCESS and TRANACT. Refer to the LIMITS statement, section 6.

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.

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 six alphanumeric characters, terminated by a period.

```
      Example 1                Example 2

          .                        .
     GOTO,1WX2.              REQUEST(TAPE1)
          .                        .
          .                        .
          .                        .
          .                  GOTO,REQUEST.
   1WX2,REQUEST(TAPE1)             .
          .                  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. All options are order-independent.

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:

1. Searching for a local file, lfn
2. Searching the system library for lfn
3. 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

DISPLAY STATEMENT

The DISPLAY statement determines the current subsystem or evaluates an expression and displays the result in the dayfile. Numeric results are displayed in both octal and decimal formats. Expression evaluation is significant only to 23 bits. Therefore, the octal representation of a negative number may be incorrect.

The statement format is:

```
    DISPLAY(SS)
          or
    DISPLAY(expression)
        expression          Any legal expression
```

Example 1:

```
    DISPLAY(SS)
```

If the BASIC subsystem is currently in use, the preceding statement inserts the following message into the dayfile:

```
    BASIC
```

Example 2:

```
    DISPLAY((R1+R3) * R2)
```

If R1=5, R2=8, and R3=3, this statement inserts the following data in the dayfile.

```
    64          100B
```

Both decimal and octal values are displayed.

SET STATEMENT

The SET statement allows the user to specify a subsystem or to set software registers to control the flow of a job. These registers are useful when designing a multipurpose procedure file. They also can be used to select a particular option in the procedure file. These software-defined registers are kept in the job control area and are preserved for the duration of the job. The control register specified in the control statement is set to the value of the expression supplied. This register can be R1, R2, R3, or EF (refer to Symbolic Names earlier in this section). The R registers are 18-bit quantities whereas the error flag (EF) is a 6-bit quantity. Excess bits are ignored.

The statement format is:

```
    SET(Ri=expression)
          or
    SET(EF=expression)
          or
    SET(SS=ssname)
        Ri          Software-defined register 1, 2, or 3
        EF          An additional register
        expression  Any legal expression
        ssname      Any legal SS subsystem name
```

Example:

This example illustrates the use of the SET statement to control execution of an object program. Because register Rl is set to 1 when file ABC is called, the object program is not executed.

```
SET(R1=1)
CALL(ABC)
FTN.
IF(R1=1) GOTO, 3.
REQUEST(TAPEI)
LGO.
3, REWIND(TAPEI)
     .
     .
     .
```

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 op ssname)stmt.
          or
IF(SS op ssname expression) stmt.
```

expression	Any legal expression
stmt	Any legal control statement
op	One of the operators: = .EQ. ≠ .NE.
ssname	Any legal SS subsystem name

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.

Example 2:

```
IF(SS.EQ.BASIC.AND.OT=TXO.AND.R1=1)GOTO,100.
SET(SS.EQ.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)
Names with true/false values	
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
DJ	844-41/44 Disk Storage Subsystem
DP	Distributive Data Path to ECS
LP	512 or 580 Line Printer
LQ	512 Line Printer
LR	580-12 Line Printer
LS	580-16 Line Printer
LT	580-20 Line Printer
MD	841 Multiple Disk Drive
MS	Mass Storage
MT	Magnetic Tape Drive (7-track)
NE	Null equipment
NT	Magnetic Tape Drive (9-track)
TT	Time-Sharing Multiplexer
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,MD.AND.PM))GOTO,200.
```

If BETA is a file on an 841 Multiple Disk Drive and it is a direct-access permanent file, processing goes 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 was used to call procedure file A

```
CALL(A,RENAME(2XY=2,T=TAPE))
```

the IF statement in A

```
IF(NUM(2XY))GOTO,1S.
```

would be evaluated as true, and control would transfer to 1S.

However, the statement

```
    IF(NUM(T))GOTO,1S.
```

would be evaluated as false, and control would pass 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
COPBR(,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, section 3.

TIME-SHARING COMMANDS

The following commands are intended for use only by time-sharing origin jobs but included here for their use in procedure files. For additional information about these commands, refer to the Time-Sharing User's Reference Manual.

ASCII STATEMENT

The ASCII control statement specifies that all subsequent operations are to be done in ASCII character set mode.

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 character set mode of the terminal.

The control statement format is:

```
CSET(m)
```

m — Current terminal character set mode; m may be one of the following.

- ASCII — Set ASCII character set mode; escape code processing is enabled
- NORMAL — Set normal character set mode; 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.

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 described in section 11.

- 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

  Program library utility control statements

  System utility control statements

  Loader control control statements †

- Product Set Control Statements

  The product set control statements call the various products available under NOS (refer to section 11).

CONTROL STATEMENT FORMAT

All control statements may consist of from 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 section 4.

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, it will not be 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 it is assumed that 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 installation handbook for a description of the SC directive.

† Refer to the CDC CYBER Loader Reference Manual.

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. After the fourth field or the third field if no parameters are present, there must be a valid terminator character.

The following is a comparison of the operating system and product set formats (refer to section 11 for a list of control statements processed in 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. Valid separators are + - " / = , (and any other character with a display code value greater than 44_8 except *) $. and blank.
2. Valid terminators are .)	2. Valid terminators are .)
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 control statements for loader control.	3. Any parameter field that includes characters other than letters, numbers, and the * must be expressed as a literal.
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.

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.

7. File names are 1 to 7 alphanumeric characters.

8. Not NOS/BE compatible

Product Set Format

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. File names beginning with a numeric character are illegal.

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, 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 control statements are processed in the following manner: parameters are extracted from the control statement and stored in the user's field length beginning at ARGR (RA+2) through RA+n (n cannot exceed 63_8). † The total number of parameters stored in the user's field length is placed in the lower 18 bits of RA+64_8. The name of the control statement is placed in bits 18 through 59 of RA+64_8.

The control statement image, less any label or prefix field, is stored at RA+70_8. If the program being executed was loaded from the system library and has an ARG= entry point, then the entire control statement image will be present at RA+70_8. Neither the information on arguments nor the argument count, however, will be entered when ARG= is present. This entry point allows for control statements with special parameter requirements (refer to appendix F, volume 2).

An example of how the control statement

PERMIT (FILEABC, USERAAA=R, USERBBB=W)

appears in CM is illustrated.

```
                                                                 Display Code
                                      Memory                      Equivalent
                   ________________________^_______________________  _____^_____

ARGR     RA+2     0611    1405    0102    0300    0000    FILEABC
                  2523    0522    0101    0100    0054    USERAAA     =
                  2200    0000    0000    0000    0000    R
                  2523    0522    0202    0200    0054    USERBBB     =
         RA+6     2700    0000    0000    0000    0000    W
           .
           .
           .
ACTR     RA+64
CCDR     RA+70    2005    2215    1124    5606    1114    PERMIT (FIL
           .      0501    0203    0025    2305    2201    EABC USERA
           .      0101    5422    5625    2305    2202    AA=R¬USERB
         RA+73    0202    5427    5755    0000    0000    BB=W)
```

The following control statements would provide exactly the same image in CM.

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

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

JOB STATEMENT FORMAT

The first statement of the control statement record is always the job statement. The job statement may be in either order-dependent or order-independent format. When the job statement is in order-independent format, the keyword and parameter are passed with no separator character. The format for the job statement is:

† The first 100_8 words of the user's field length, from RA through RA+77_8, comprise the job communication area. Refer to appendix E, volume 2 for a description of this area.

jobname(Tt, CMfl, Pp).....cm

jobname(p, t, fl).....cm

jobname	Alphanumeric job name (1 to 7 characters) which must begin with a letter. This name identifies individual jobs being run under the same user number.
t	Central processor job step time limit in octal seconds, ranging from 1 to 77770_8. The time limit must be sufficient for completion of each job step in the job. If t is absent, the system assumes t equals 100_8 (100_8 seconds is approximately 1 minute).
fl	Maximum CM field length (storage requirement) for the job. The system rounds the value to the next highest multiple of 100_8. The field length cannot exceed: $377,700_8$ on a 198K or a 262K machine $360,000_8$ on a 131K machine $163,000_8$ on a 65K machine $61,000_8$ on a 32K machine

NOTE

The following messages are issued to the user's dayfile if validation limits are exceeded.

CM NOT VALIDATED.	The number of CM words specified on the job statement exceeds that for which the user is validated.
TL NOT VALIDATED.	The time limit specified on the job statement exceeds that for which the user is validated.

The user may be further restricted by limits placed on him by the validation file or by installation parameters. The user should consult installation personnel for restrictions based on the machine configuration and subsystems used.

In addition, RFL (the running field length for a job step) will always be zero unless the user specifies a field length with the RFL control statement (refer to section 6). Whenever RFL is zero, the system is in control of field length assignment. The MFL (maximum field length) control statement will clear any RFL value previously set with an RFL control statement (refer to section 6).

p	Priority level (octal) at which the job enters the system; $1 \leq p \leq 17_8$. This parameter is currently ignored since the system will automatically assign priorities specified by the installation parameters.

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

In addition to the regular separator characters, the * may also be used to separate parameters on the job statement.

If the order-dependent format is employed and null parameters are indicated with multiple separators, the null parameters are interpreted as zeros.

Example:

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

has the same effect as

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

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, 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.

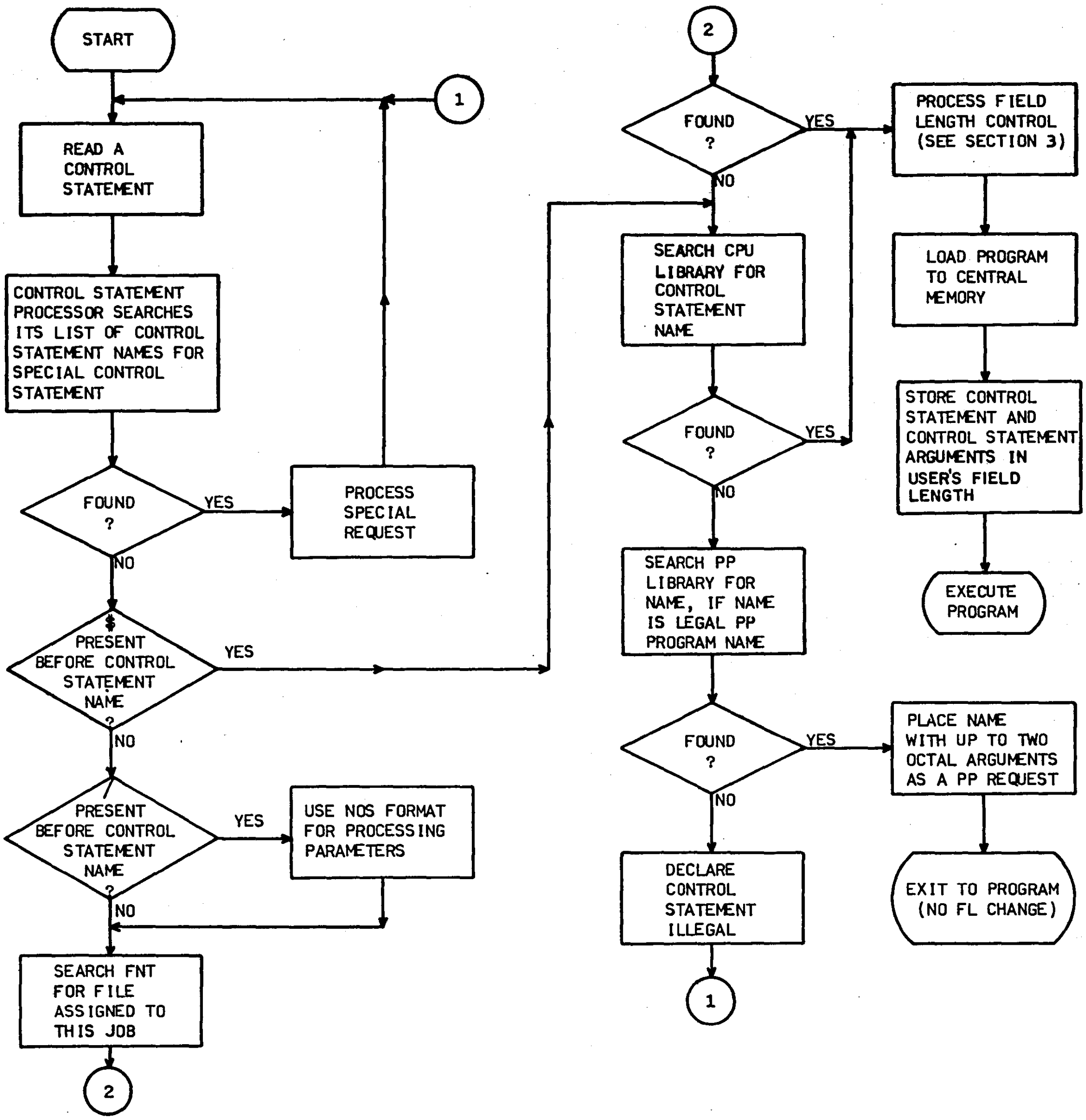

START
READ A CONTROL STATEMENT
1
CONTROL STATEMENT PROCESSOR SEARCHES ITS LIST OF CONTROL STATEMENT NAMES FOR SPECIAL CONTROL STATEMENT
FOUND ?
YES
PROCESS SPECIAL REQUEST
NO
$ PRESENT BEFORE CONTROL STATEMENT NAME ?
YES
NO
/ PRESENT BEFORE CONTROL STATEMENT NAME ?
YES
USE NOS FORMAT FOR PROCESSING PARAMETERS
NO
SEARCH FNT FOR FILE ASSIGNED TO THIS JOB
2
2
FOUND ?
YES
PROCESS FIELD LENGTH CONTROL (SEE SECTION 3)
NO
SEARCH CPU LIBRARY FOR CONTROL STATEMENT NAME
FOUND ?
YES
NO
SEARCH PP LIBRARY FOR NAME, IF NAME IS LEGAL PP PROGRAM NAME
FOUND ?
YES
PLACE NAME WITH UP TO TWO OCTAL ARGUMENTS AS A PP REQUEST
NO
DECLARE CONTROL STATEMENT ILLEGAL
1
LOAD PROGRAM TO CENTRAL MEMORY
STORE CONTROL STATEMENT AND CONTROL STATEMENT ARGUMENTS IN USER'S FIELD LENGTH
EXECUTE PROGRAM
EXIT TO PROGRAM (NO FL CHANGE)

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. For further discussion of possible error conditions, refer to section 3 of this manual.

The following sequence of control statements illustrates this exit processing.

```
JOBCCC.
USER(SMITH22,SM)
CHARGE(55A19)
NOEXIT.
GET(A,B)
ONEXIT.
ATTACH(MASTER/M=W)
SKIPEI(MASTER)
COPYBF(A,MASTER)
COPYB(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 will continue 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
CHARGE
COMMENT
CTIME
DAYFILE
ENQUIRE
EXIT
LDI
LENGTH
LIMITS
MFL
MODE
NOEXIT
NORERUN
OFFSW
ONEXIT
ONSW
PASSWOR
RERUN
RESOURC
RFL
ROLLOUT
RTIME
SETASL
SETJSL
SETPR
SETTL
STIME
SUBMIT
SUI
SUMMARY
SWITCH
USECPU
USER

The user must have specific validation parameters set to use LDI, PASSWOR, 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 also 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, and DAYFILE statements. The COMMENT statement allows the user to provide his own 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 will read 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 messages issued to the user's dayfile, refer to Job Completion, section 3.

COMMENT STATEMENT

The COMMENT statement is used to enter 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

```
*comment
```

format is used, the * must appear in column 1.

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 (seven 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 to the user identification and priority information.

Example:

```
USER NUMBER          DLH2500
USER INDEX HASH      AKQA
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.

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

CONTROL STATEMENT(S).

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

Option Description

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. These statistics are factors that make up the SRU.

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 amount of resources still available to the user.

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
```

Option — Description

jobname — Last three characters of the name assigned by the system to a remote batch job that has been initiated with the SUBMIT, ROUTE, or LDI statement. When this parameter is specified, the status of the remote batch 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. It is only possible to obtain 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.

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 controls tatements later in this section and to the description of exit processing in section 5.

The control statement format is:

```
    EXIT.
```

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 coniunues until an EOI or double EOF is encountered. The jobs submitted are gatch origin type jobs.

The control statement format is:

```
    LDI(ifn,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 will be 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 will be 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 attached 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 indirect access permanent files the user can have in each catalog. This limit applies to each catalog being accessed (main, public auxiliary, or private auxiliary).

CS Maximum number ot 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† Specifies the 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

MS Maximum number of mass storage PRUs the user is allowed to additionally allocate via his job

DF Maximum number of MESSAGE requests the user 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

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

Field Description

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 1, 3, and 4).

Option Specifies

1 User can change his password.

2 User can use the privileged time-sharing commands. ††

3 User is allowed to create direct access files.

4 User is allowed to create indirect access files.

5 User can have system origin (SYOT) capability from 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.

6 User can access/create library files.

7 User can assign nonallocatable devices. A nonallocatable device is a magnetic tape unit, card reader, card punch, or line printer. Refer to File Management Control Statements in section 7 for further information.

8 User is allowed to access the system without supplying his assigned charge and project numbers.

9 User can define, save, and replace files on auxiliary devices.

10 User can access special transaction functions.

11 Allows no terminal timeout.

12 User has special accounting privileges. †††

13 Allows use of the system control point (SCP) facility.

The octal value listed for AW corresponds to the preceding options where bit 0 is option 1, bit 1 is option 2, and so on. For example, if the access word listed were:

```
AW=00000000000000000215
```

the user would be validated for options 1, 3, 4, and 8.

†Not currently used by the system but provided for future expansion of validation control.

††For further information about privileged time-sharing commands, refer to the operator's guide.

†††Refer to part IV, section 1 of the NOS Installation Handbook for a description of special user's accounting privileges.

The LIMITS statement is equivalent to the OP=I option of MODVAL. 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 a job step. The control statement format is:

```
    MFL(nnnnnn)

        nnnnnn              Field length (octal)
```

The parameter nnnnnn sets an upper bound 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 the field length required by the utility (CONTROL) that processes MFL. The field length required by CONTROL is 400B.

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 will continue to determine field lengths until another RFL control statement or SETRFL macro is encountered.

MODE STATEMENT

The control statement format is:

MODE(m, n)

m — CPU program error exit mode ($0 \le m \le 7$)

n[†] — CPU hardware error exit mode ($0 \le n \le 7$)

[†] Applicable to CDC CYBER 170 series only.

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.

The following values can be supplied for n.

n	CPU Hardware Error Exit Mode
0	Disable hardware exit mode; no selection made
1	ECS flag register operation parity error
2	CMC input error
3	ECS flag register operation parity error or CMC input error
4	CM data error
5	ECS flag register operation parity error or CM data error
6	CMC input error or CM data error
7	ECS flag register operation parity error, CMC input error or CM data error. If no n mode is selected, the system assumes n=7.

It is recommended that the user always specify n=7. If any hardware exits occur, he should contact a customer engineer or on-site analyst.

The MODE statement is used to define 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. Note that 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 will occur for either a mode 1 or mode 4 error condition. Refer to Error Control, section 3 and to the CDC 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 error does not cause job to 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.

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; $1 \le s_i \le 6$. If $s_i=0$ is specified, all sense switches are cleared.

The system stores the sense switch settings in the user's control point area and copies them to RA for use by the central program. The system operator can change these settings by console command.

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 a 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 control point area and copies them to RA for use by the central program. The system operator can change these settings by console command.

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

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

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 necessary in any job that uses more than one tape or pack concurrently in order to prevent deadlocks with other jobs which may need the same resources.

The control statement format is:

RESOURC(rt_1=u_1, rt_2=u_2,..., rt_n=u_n)

rt_i Resource type:

MT	Magnetic Tape Unit (7-track)
NT	Magnetic Tape Unit (9-track)
DIi	844-21 Disk Storage Subsystem ($1 \leq i \leq 8$)
DJi	844-41/44 Disk Storage Subsystem ($1 \leq i \leq 8$)
MDi	841 Multiple Disk Drive ($1 \leq i \leq 8$)

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.

The system manages the use of tape units and disk packs in such a way as to prevent deadlocks from occurring. A deadlock would occur if the system, by assigning a tape unit or pack to one job, prevented 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 relinquished its assigned unit.

The system prevents such situations by requiring that a RESOURC control statement be included in any job that uses more than one tape or pack concurrently. 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.

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, 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.

Thus, in the previous example, a RESOURC statement would be required in both jobs. The information supplied by the statements would enable the system to anticipate the deadlock situation and roll out job B until job A no longer needed both units.

[†] 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.

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

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.

The system manages resources by keeping totals of the number of scheduled units and assigned units. Each total can vary during job processing. A user can increase the number of scheduled units by returning all files attached to his job residing on resource units not currently needed and then scheduling the required number of units with another RESOURC statement. He can decrease the number of scheduled units by including RETURN statements or additional RESOURC statements.

In the following job, for 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)
RESOURC(NT=2)         )      At some time during this phase of processing,
                      |      the job will require two 9-track tape units.
     .                |
     .                }
     .                |
     .                )
RESOURC(MD1=2,NT=1)   )      During this phase, the job will require two 841
     .                |      Multiple Disk Drives and one 9-track tape unit.
     .                }      The NT=1 entry decreases the number of scheduled
     .                |      tape units from two to one.
     .                |
-EOI-                 )
```

If the user decreases the total to less than the number of currently assigned units or increases the total to a point where a deadlock would occur, the system issues an error message to the user's dayfile and aborts his job.

NOTE

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

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 sharable, 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 non-sharable 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 a job step when neither the routine for processing that step nor a loader table specifies a field length.

The control statement format is:

RFL(nnnnnn)

nnnnnn	Field length (unless decimal is specified, octal is assumed)

If the field length is specified in decimal (number contains an 8 or 9 or has a post-radix of D), it is converted to octal and rounded up to the nearest 100_8. The value of nnnnnn cannot exceed the value 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 Job Control, section 3).

ROLLOUT STATEMENT

The ROLLOUT control statement requests that the user's job be rolled out and all memory assigned to the job released.

The control statement format is:

```
ROLLOUT.
```

The user's job is entered into the rollout queue and is rescheduled by the system.

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 dèadstart.

The control statement format is:

```
RTIME.
```

SETASL STATEMENT

The SETASL control statement allows the user to specify a new account block SRU limit.

The control statement format is:

```
SETASL(s)
```

s	Account block SRU limit in units (maximum is 77777_8, which is infinite)

The account block SRU limit is the number of SRUs that may be accumulated by the job before the system issues the error message:

```
ACCOUNT BLOCK LIMIT.
```

Each user and charge/project number is validated for a maximum SRU limit. If the user attempts to set the account block SRU limit above this limit, the following message is issued.

```
SL NOT VALIDATED.
```

If $1 \leq s \leq 77777_8$ is not satisfied, the following message is issued.

```
ILLEGAL USER ACCESS.
```

The parameter s represents the maximum SRU accumulation between CHARGE statements or between one CHARGE statement and the end of the job. If a CHARGE statement is not required, s represents the maximum SRU accumulation from the USER statement to the end of the job.

The user may not set the account block SRU limit to a value less than the current job step SRU limit. An attempt to do so will result in the message:

```
JOB STEP EXCEEDS ACCOUNT BLOCK.
```

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.

SETJSL STATEMENT

The SETJSL control statement allows the user to specify the job step SRU limit for each remaining step of his job.

The control statement format is:

```
SETJSL(s)
```

s	Job step SRU limit in units (maximum is 77777_8, which is infinite)

The job step SRU limit is the number of SRUs that may be accumulated by a single job step before the system issues the error message:

```
JOB STEP LIMIT.
```

The job step SRU limit may not exceed the account block SRU limit (the number of SRUs which may be accumulated by the job). If this is attempted, the following message is issued.

```
JOB STEP EXCEEDS ACCOUNT BLOCK.
```

If $1 \leq s \leq 77777_8$ is not satisfied, the following message is issued.

```
ILLEGAL USER ACCESS.
```

SETPR STATEMENT

The SETPR control statement allows the user to specify a new CPU priority for his job.

The control statement format is:

```
SETPR(p)
```

p — Priority, $1 < p < 70_8$; if p exceeds that for which the user is validated, it is reduced to that value.

The CPU priority controls the assignment of the CPU to active jobs. If the CPU priority is lower than that of other jobs, the job is assigned to the CPU only when jobs of a higher priority do not need it. The user is validated for a maximum CPU priority. He cannot request a level that exceeds this value or 70_8 (the maximum CPU priority).

SETTL STATEMENT

The SETTL control statement allows the user to specify a new CPU time limit for subsequent job steps.

The control statement format is:

```
SETTL(t)
```

t — Central processor job step time limit in octal seconds (maximum is 77777_8); t is accurate to the nearest second. If an 8 or 9 appears in the specification, it is interpreted as decimal.

The CPU time limit is the amount of time (in seconds) that a job step is allowed to use the CPU before the error message

```
TIME LIMIT.
```

is issued by the system.

The user is validated for a maximum job step time limit. If this is exceeded or $1 \leq t \leq 77777_8$ is not satisfied, the following message is issued.

```
TL NOT VALIDATED.
```

If t is between 77770_8 and 77777_8, the time limit is infinite. The user cannot set a time limit greater than that for which he is validated.

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.
```

SUBMIT STATEMENT

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 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 disposed to local batch queue, but is dropped at job termination (default value for time-sharing origin jobs)
- E — Job output is disposed to Export/Import 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 submit file lfn contains a batch job submitted to the system for processing. The reformatting directives described in this section are provided to aid the user in preparing the submit file. When the SUBMIT statement is processed, the submit file can be reformatted according to the directives that appear in the file.

The number of jobs that the user can have in the system concurrently is dependent on the individual validation limit (refer to the DB field of the LIMITS control statement in this section). If this limit is exceeded, the following message is issued to his dayfile.

```
    TOO MANY DEFERRED BATCH JOBS.
```

Each line in the submit file preceded by an escape character is recognized by the system as a reformatting directive. The escape character to be used must be 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 that is to be reformatted before processing should be

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

where ln1, ln2, and ln3 are optional line numbers.

The SEQ and NOSEQ directives are used to determine, during reformatting, if the submit file will contain leading line numbers. Therefore, it is a simple matter to include line numbers on the entire submit file and specify which line numbers are to be removed during reformatting. This is especially useful if the submit file contains a BASIC program where line numbers are a requirement of the language.

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.

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.

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

Indicates transparent mode. When the system encounters this directive, it checks the next line of the submit file for an additional directive. If one exists, it is processed and the next line is checked. This continues until a line that is not a reformatting directive is encountered. Transparent mode is then selected and all directives that exist on subsequent lines are ignored until an internal EOR or EOF is encountered (this pertains only to EOR and EOF marks that currently exist, not cEOR and cEOF directives). The cPACK and cNOPACK directives determine if the internal EOR or EOF mark will be retained. The line following the internal EOR or EOF mark is then checked for a reformatting directive. If one exists, it is processed and the following line is checked. All directives are processed until a line that does not contain a reformatting directive is encountered. This causes transparent mode to be reset unless a cNOTRANS directive was encountered. This process continues until either the end of the submit file is reached or until a cNOTRANS directive following an internal EOR or EOF is encountered.

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. Note that 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 will be 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. Note, however, that 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 will be 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 in the submit file or an internal EOR or EOF mark.

cREAD,lfn

Requests that the system read the entire contents of the specified file, lfn, and insert that file in place of the cREAD directive in the submit file, during reformatting. 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 and EOF marks will be removed. If the cNOPACK directive is in effect, all internal EOR and EOF 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 will also be checked for a reformatting directive. Any directives contained in the file, except another cREAD, will be processed. The cREAD directive cannot be nested. In addition, any directives in effect before the cREAD directive is processed will 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 or EOF in the file being read will be processed.

If the file to be read is a binary file, it is recommended that the cTRANS directive be used. This is to ensure that binary data will not be 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 will be used to recognize all subsequent reformatting directives until further change.

There is no restriction on the maximum number of characters per line for transparent mode. For all other modes, no line can exceed 150 (6-bit) characters.

If the user determines that an error occurred during processing of his job, he may reference a listing of the user's dayfile as an aid in identifying the cause of the error. The user's dayfile contains a record of the job processing activity and is disposed to the local batch queue or the Export/Import queue for printing when the job is terminated. However, all output is normally dropped at job termination when a batch job image is submitted from a time-sharing terminal. This includes the dayfile output as well as the job output. In this event, the user can make provisions within his job to save the contents of the dayfile if an error in processing 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)
```

If the submitted job contains an illegal USER statement, the job entering the SUBMIT statement is aborted (no exit processing), and 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 will be decremented accordingly.

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

SIUN, usernum.

Terminal users will be immediately logged off with no dayfile message. For further information concerning use of the SUBMIT statement from a time-sharing terminal, refer to the Time-Sharing User's Reference Manual.

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:

$\mathrm{SWITCH}(s_1, s_2, \ldots, s_n)$

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

The system stores the sense switch settings in the control point area and copies them to RA for use by the central program. The system operator can change these settings by console command.

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 CDC CYBER 73-2x, 74-2x, 6500, 6700, or CDC 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 utilizes the USER control statement to determine if the programmer is a legal user, 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 of jobs entering the system via an LDI or SUBMIT statement. 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	A 1- to 7-character alphanumeric password
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 will also be aborted, the security count decremented, and the preceding messages issued if an illegal or invalid USER statement is detected at any time, regardless of whether secondary USER statements are enabled or disabled. In all cases, terminal users will be immediately logged off with no dayfile message issued to the terminal.

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

```
ILLEGAL USER NUMBER - CONTACT SITE OPR.
```

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

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 had to be 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.

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

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 devices.

FILE MANAGEMENT CONTROL STATEMENTS 7

The file management control statements enable the user to manipulate files attached to his job. The control statements included in this category are:

ASSIGN
BKSP
CATALOG
CLEAR
COMMON
CONVERT
COPY
COPYBF
COPYBR
COPYCF
COPYCR
COPYEI
COPYSBF
COPYX
DISPOSE
DOCMENT
EVICT
GTR
LIBEDIT
LIBGEN
LIST80
LOCK
LO72
NEW
OUT
PACK
PRIMARY
RENAME
REQUEST
RESEQ
RETURN
REWIND
ROUTE
SETID
SKIPEI
SKIPF
SKIPFB
SKIPR
SORT
STAGE
TDUMP
UNLOAD
UNLOCK
VERIFY
VFYLIB
WRITEF
WRITER

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. Refer to the ASSIGN and REQUEST statements in this section and Tape Management control statements in section 10 for a more detailed description.

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:

$\text{ASSIGN(nn, lfn, }\begin{Bmatrix}\text{CK}\\ \text{CB}\end{Bmatrix}\text{)}$

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
CP	415 Card Punch
CR	405 Card Reader
DE	Extended Core Storage
DI	844-21 Disk Storage Subsystem
DJ	844-41/44 Disk Storage Subsystem
DP	Distributive Data Path to ECS
LP	512 or 580 Line Printer
LQ	512 Line Printer
LR	580-12 Line Printer
LS	580-16 Line Printer
LT	580-20 Line Printer
MD	841 Multiple Disk Drive
MS	Mass Storage Device
NE	Null Equipment
TT	Time-Sharing Multiplexer[††]

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.

Before performing the assignment, the system issues a RETURN on lfn. Any job can assign a file to MS, and any time-sharing origin job can assign a file to TT. However, to assign any other devices, the job must be of system origin or the user must be validated for system origin privileges.[†††]

If the user attempts to perform an assignment for which he is not validated, the job is aborted and the following message is issued to the user's dayfile.

ILLEGAL USER ACCESS.

[†] Contact installation personnel for a list of EST ordinals.

[††] This device type applies only to time-sharing origin jobs.

[†††] Refer to LIMITS control statement, section 6.

In addition, to assign a file to any nonmass storage device except device type TT, the user must be validated to use nonallocatable devices. If the user does not have this validation or the device is not available, the system aborts the job.

The user should not normally assign any nonallocatable devices to his job. While it is possible to assign a central site card reader, line printer, or card punch directly online to the user's job, only a subset of the capabilities of local batch input/output are available through this method of access. Also, there is no need to assign nonallocatable devices to local files named OUTPUT, PUNCH, P8, or PUNCHB and any other local file disposed to an output queue because these files are always processed upon job completion.

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. Informative and error 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 causes output that would normally be written on XYZ to be printed at the terminal instead.

Example 3:

```
    ASSIGN(MD, ABC)
```

This statement assigns file ABC to an 841 Multiple 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, 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

```
    BSKP(FILE1)
```

request would reposition FILE1 to the beginning of the third record. The system does not backspace past the beginning of information (BOI). 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.

CATALOG STATEMENT

The CATALOG control statement requests a listing of information about each record in a specified file.

The control statement format is:

CATALOG(lfn, $p_1, p_2, \ldots, p_n$)

lfn — Name of the file to be cataloged

p_i — May be one of the following.

- N=0 — Catalog until an empty file is encountered
- N=x — Catalog x files; default is N=1
- N — Catalog to end of information
- L=fname — Specifies the name of the file to receive output; if this parameter is omitted, the system assumes L=OUTPUT
- U — Select user library list (not given unless selected)
- D — Suppress all comment fields; suppress all page headings after the initial page heading for each individual file
- R — Rewind lfn before and after cataloging
- CS — Suppress character set list for OPL (old program library) and OPLC (old program library common deck) type records.

The listing for each file of a multifile set 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 includes:

- Number of the record cataloged
- Record name from the first word of the record or the second word of the prefix (77) table, if present
- Record type (list of valid record types follow this list)
- Length (less 77 table length) in words printed as an octal number
- A checksum (not including the 77 table)
- Dates and comments in 77 table, if present
- Character set mode for OPL/OPLC type records (unless suppressed by CS option)

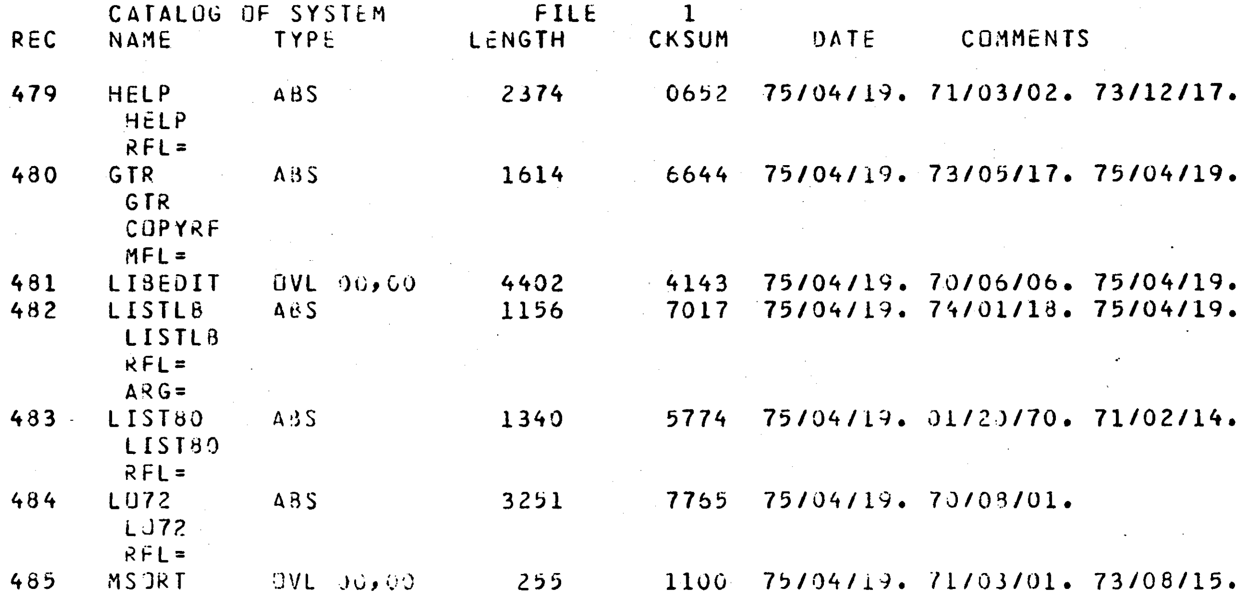
Type may be one of the following.

ABS	Multiple entry point overlay.
CAP	Capsule loader record (supported by CDC CYBER Loader 1.3)
OPL	Modify old program library deck
OPLC	Modify old program library common deck
OPLD	Modify old program library directory
OVL	Central processor overlay
PP	6000 series peripheral processor program
PPU	7600 peripheral processor program
REL	Relocatable central processor program
TEXT	Unrecognizable as a program
ULIB	User library program

Entry points are listed for REL and ABS format records. The entire record is listed for TEXT format records if the name of the record begins with CMRDECK, CMRDC, IPRDECK, IPRDC, LIBDECK, or LIBDC. The first line is listed for TEXT format records if the name of the record begins with OVERLAY. Correction identifiers and their YANK status (refer to the Modify Reference Manual) are listed for OPL and OPLC records.

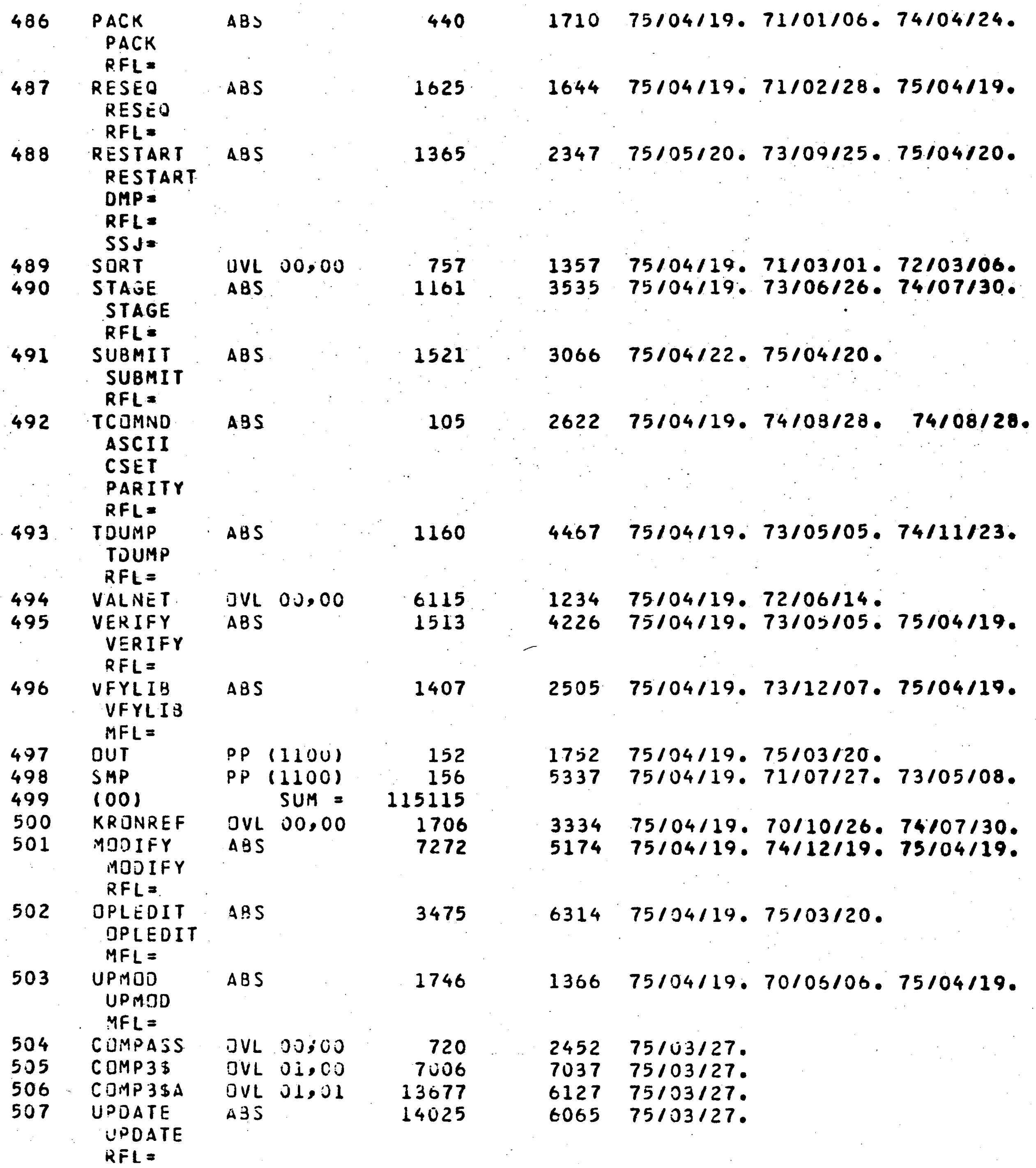
A ULIB format record suppresses listing of records in the library unless the U option is specified on the control statement. Zero-length records cause the length since the last zero-length record to be listed. EOFs cause the length since the last EOF to be listed.

Figure 1-7-1 illustrates a portion of the catalog of SYSTEM.



```
        CATALOG OF SYSTEM             FILE       1
REC     NAME        TYPE            LENGTH     CKSUM      DATE       COMMENTS

479     HELP        ABS               2374      0652   75/04/19. 71/03/02. 73/12/17.
         HELP
         RFL=
480     GTR         ABS               1614      6644   75/04/19. 73/05/17. 75/04/19.
         GTR
         COPYRF
         MFL=
481     LIBEDIT     OVL  00,00        4402      4143   75/04/19. 70/06/06. 75/04/19.
482     LISTLB      ABS               1156      7017   75/04/19. 74/01/18. 75/04/19.
         LISTLB
         RFL=
         ARG=
483     LIST80      ABS               1340      5774   75/04/19. 01/20/70. 71/02/14.
         LIST80
         RFL=
484     LO72        ABS               3251      7765   75/04/19. 70/08/01.
         LO72
         RFL=
485     MSORT       OVL  00,00         255      1100   75/04/19. 71/03/01. 73/08/15.
```

Figure 1-7-1. Sample Page of Catalog of SYSTEM



```
486     PACK      ABS              440        1710  75/04/19. 71/01/06. 74/04/24.
         PACK
         RFL=
487     RESEQ     ABS             1625        1644  75/04/19. 71/02/28. 75/04/19.
         RESEQ
         RFL=
488     RESTART   ABS             1365        2347  75/05/20. 73/09/25. 75/04/20.
         RESTART
         DMP=
         RFL=
         SSJ=
489     SORT      OVL  00,00       757        1357  75/04/19. 71/03/01. 72/03/06.
490     STAGE     ABS             1161        3535  75/04/19. 73/06/26. 74/07/30.
         STAGE
         RFL=
491     SUBMIT    ABS             1521        3066  75/04/22. 75/04/20.
         SUBMIT
         RFL=
492     TCOMND    ABS              105        2622  75/04/19. 74/08/28.  74/08/28.
         ASCII
         CSET
         PARITY
         RFL=
493     TDUMP     ABS             1160        4467  75/04/19. 73/05/05. 74/11/23.
         TDUMP
         RFL=
494     VALNET    OVL  00,00      6115        1234  75/04/19. 72/06/14.
495     VERIFY    ABS             1513        4226  75/04/19. 73/05/05. 75/04/19.
         VERIFY
         RFL=
496     VFYLIB    ABS             1407        2505  75/04/19. 73/12/07. 75/04/19.
         VFYLIB
         MFL=
497     OUT       PP  (1100)       152        1752  75/04/19. 75/03/20.
498     SMP       PP  (1100)       156        5337  75/04/19. 71/07/27. 73/05/08.
499     (00)          SUM =     115115
500     KRONREF   OVL  00,00      1706        3334  75/04/19. 70/10/26. 74/07/30.
501     MODIFY    ABS             7272        5174  75/04/19. 74/12/19. 75/04/19.
         MODIFY
         RFL=
502     OPLEDIT   ABS             3475        6314  75/04/19. 75/03/20.
         OPLEDIT
         MFL=
503     UPMOD     ABS             1746        1366  75/04/19. 70/06/06. 75/04/19.
         UPMOD
         MFL=
504     COMPASS   OVL  00,00       720        2452  75/03/27.
505     COMP3$    OVL  01,00      7006        7037  75/03/27.
506     COMP3$A   OVL  01,01     13677        6127  75/03/27.
507     UPDATE    ABS            14025        6065  75/03/27.
         UPDATE
         RFL=
```

Figure 1-7-1. Sample Page of Catalog of SYSTEM (Contd)

CLEAR STATEMENT

The CLEAR control statement releases all the user's current working files.

The control statement format is:

```
CLEAR.
```

If a primary file exists, only the file name is retained; information within the file is purged. The empty file remains available as the primary file.

COMMON STATEMENT

The COMMON control statement is used to either create or access a library type file.

The control statement format is:

COMMON(lfn_1, lfn_2, . . . , lfn_n)

lfn   Logical file name

The user must be validated to access/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 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, . . . , 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); $1 \le n \le 500$. 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

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 table 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 causes the first file specified to be copied to the second file.

The control statement format is:

COPY(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 begins 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 SI, S, or L format tape is performed in coded rather than binary mode.

The copy begins at the current position of both files, unless the x parameter is specified, and continues until an empty file (a double EOF) or EOI is encountered in lfn_1. If the copy is terminated by a double EOF, the second EOF is detected but is not transferred to lfn_2. That is, if the files are not rewound after the copy (x parameter not specified), file lfn_1 is positioned after the second EOF and lfn_2 after the first EOF.

When a COPY control statement operates with B or E format magnetic tapes, a specific frame count (FC) is required to ensure logical coincidence between the original and the copy (refer to ASSIGN and LABEL Statements, section 10). For disk-to-tape and tape-to-disk copies, FC must equal 640, and for tape-to-tape copies, the FC counts for both tapes must be equal.

The COPY statement may produce unpredictable results when copying S, L, and F format tapes. The user should check these formats in section 10 before using them with this control statement.

COPYBF STATEMENT

The COPYBF control statement causes a specified number of binary files to be copied from one file to another.

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 SI, S, or L format tape is performed in coded rather than binary mode.

The copy begins at the current position of lfn_1. 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 on lfn_2, and the operation terminates.

When a COPYBF control statement operates with B or E format magnetic tapes, a specific frame count (FC) is required to ensure logical coincidence between the original and the copy (refer to ASSIGN and LABEL Statements, section 10). For disk-to-tape and tape-to-disk copies, FC must equal 640, and for tape-to-tape copies, the FC counts for both tapes must be equal.

The COPYBF statement may produce unpredictable results when copying S, L, and F format tapes. The user should check these formats in section 10 before using them with this control statement.

COPYBR STATEMENT

The COPYBR control statement causes a specified number of binary records to be copied from one file to another.

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 SI, S, or L format tape is performed in coded rather than binary mode.

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, n records are skipped but no data transfer occurs. If the EOI is encountered before the record count is satisfied, an EOF is written on lfn_2, and the operation terminates.

The COPYBR statement may produce unpredictable results when copying S, L, and F format tapes. The user should check these formats in section 10 before using them with this control statement.

COPYCF STATEMENT

The COPYCF control statement directs the system to copy a specified number of files from one file to another.

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, n files are skipped but no data transfer occurs. If the EOI is encountered before the file count is satisfied, an EOF is written on lfn_2, and the operation terminates. COPYCF reformats the file into line images if it is blocked in greater than lchar blocks.

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.

If n is illegal or zero, the following message is issued.

ILLEGAL COUNT.

The COPYCF statement may produce unpredictable results when copying S, L, and F format tapes. The user should check these formats in section 10 before using them with this control statement.

COPYCR STATEMENT

The COPYCR control statement directs the system to copy a specified number of records from one file to another.

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, n records are skipped but no data transfer occurs. 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.

The COPYCR statement may produce unpredictable results when copying S, L, and F format tapes. The user should check these formats in section 10 before using them with this control statement.

COPYEI STATEMENT

The COPYEI control statement directs the system to copy one file to another.

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 SI, S, or L format tape is performed in coded rather than binary mode.

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 Data Formats, section 10).

When a COPYEI control statement operates with B or E format magnetic tapes, a specific frame count (FC) is required to ensure logical coincidence between the original and the copy (refer to ASSIGN and LABEL Statements, section 10). For disk-to-tape and tape-to-disk copies, FC must equal 640, and for tape-to-tape copies, the FC counts for both tapes must be equal.

The COPYEI statement may produce unpredictable results when copying S, L, and F format tapes. The user should check these formats in section 10 before using them with this control statement.

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.

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 will be lost. A page eject character is inserted at the beginning of each logical record (refer to section 9 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.

The COPYSBF statement may produce upredictable results when copying S, L, and F format tapes. The user should check these formats in section 10 before using them with this control statement.

COPYX STATEMENT

The COPYX control statement enables the user to specify certain conditions when copying logical records.

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 (double EOF or EOI)
name	Copy all records up to and including record of specified name (record name is first seven characters of record)
type/name	Copy all records up to and including record of specified type and name (refer to CATALOG control statement 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 SI, S, or L format tape is performed in coded rather than binary mode.

The COPYX routine copies logical records from file 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 lfn_1=lfn_2, the file is repositioned according to the x parameter; no data is transferred.

The COPYX statement may produce unpredictable results when copying S, L, and F format tapes. The user should check these formats in section 10 before using them with this control statement.

DISPOSE STATEMENT†

The DISPOSE control statement is used to release specified files to the proper output queues.

The control statement format is:

DISPOSE(lfn_1=q_1,lfn_2=q_2,...,lfn_n=q_n/ot=usernum)

lfn_i Name of the file to be disposed

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 (Export/Import)

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, ..., 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 C, volume 2 for a detailed explanation of the documentation standards followed. It also contains examples of external and internal documentation for program COPYB.

EVICT STATEMENT

The EVICT control statement releases file space for a specified file(s) but does not release file attachment to the job.

The control statement format is:

EVICT(lfn_1, lfn_2, . . . , lfn_n)

lfn_i Name(s) of the file(s) to be evicted

The operation that EVICT performs depends on the file type. For permanent files, all file space except the first track is released, job attachment remains, and an EOI is written on the first sector of the first track. For all other file types, file space is released and job attachment remains. Also, all files for which write lockout is set are returned to the system. An EVICT of a tape file performs the same function as a RETURN except that EVICT cannot be used to decrease the number of tape units scheduled via the RESOURC statement.

GTR STATEMENT

The GTR control statement provides directives for specifying certain records to be copied from one file to another.

The control statement format is:

GTR(lfn$_1$, lfn$_2$, D, NR, S)selection directives

The parameters must be entered in the order shown; they are defined as follows:

lfn$_1$ — File which is searched for the selected records; if this parameter is omitted, file OLD is assumed.

lfn$_2$ — File on which the selected records are written; if this parameter is omitted, file LGO is assumed.

D — If specified, a directory record (OPLD type) is written at the end of lfn$_2$. In this case, lfn$_2$ must be a mass storage file.

This parameter has special meaning for ULIB type records, as follows:

> If D is omitted, the first record of the user library, that is, the directory record (UPLD), is not copied to lfn$_2$; the last record (OPLD type) is copied but is not altered.
>
> If D is specified, the first record of the user library (UPLD) is copied to lfn$_2$, but is not altered, and an additional record, a new directory for the file (OPLD type), is added to lfn$_2$.

NR — If specified, neither file is rewound after the operation. If not specified, both files are rewound after the operation.

S — lfn$_1$ is processed as a sequential file; no attempt is made to read a directory.

selection directives — The user can specify the record types and names that he wants retrieved; these can be:

> type/name — Retrieves record of specified type and name (refer to CATALOG control statement for a list of valid record types). The record name is the first seven characters of the record.
>
> name — Retrieves the record specified; the type is either TEXT or the type specified previously. If name=*, all records of the specified type are retrieved.
>
> 0 — Inserts a zero-length record on file lfn$_2$.
>
> type/name$_1$-name$_2$ — Retrieves records name$_1$ through name$_2$ of type specified.

GTR searches file lfn$_1$ for the records specified by the selection directives. The selected records are then copied to file lfn$_2$. If lfn$_2$ is a tape file, the selected records are copied from the current position; if lfn$_2$ resides on mass storage, the copy starts at the current EOI of the file. This is because lfn$_2$ is treated as a random file. Note that blanks are not legal between the terminator and the selection directives.

Examples of the use of this control statements are:

- GTR(SYSTEM, BIN, D)PP/*

  All records of type PP are retrieved from file SYSTEM and copied to file BIN. A directory is built and placed as the last record on file BIN.

- GTR(OPL, NEW, , NR)OPLC/COMCARG, 0, COMCCIO

  Record COMCARG (type OPLC) is retrieved from file OPL and written on file NEW beginning at the current EOI. Then a zero-length record is written on file NEW. Finally, record COMCCIO (also type OPLC) is retrieved from file OPL and written on file NEW at its current position. File OPL is not rewound either before or after the operation.

- GTR(SYSTEM, SYSLIB, D)ULIB/SYSLIB

  The record named SYSLIB (type ULIB) is retrieved from file SYSTEM and copied to file SYSLIB. The D parameter must be specified to copy the ULIB directory (UPLD) of a ULIB record set. If the D parameter were omitted, the UPLD record would be skipped.

LIBEDIT STATEMENT

The LIBEDIT control statement specifies directives for editing and replacing binary records on a file with records from one or more correction files.

The control statement format is:

LIBEDIT($p_1, p_2, \ldots, p_n$)

p_i Any of the following parameters in any order:

I=lfn$_1$	Directives comprise the next record on file lfn$_1$.
I=0	No directive input.
I omitted	Directives are on file INPUT.
P=lfn$_2$	File lfn$_2$ contains the old program library.
P=0	No old program library file.
P omitted	Old program library is on file OLD.
N=lfn$_3$	New program library will be written on file lfn$_3$.
N=0	Illegal; no error message is issued, if used.
N omitted	New program library will be written on file NEW.

NOTE

The new program library is evicted prior to processing (refer to EVICT Statement in this section).

L=1	Short correction listing (includes only directives, modifications, and errors) on the file specified by the LO parameter.
L=0	No output is listed.
L omitted	Full correction listing is written on the file specified by the LO parameter.
LO=lfn$_4$	List output on file lfn$_4$.
LO omitted	List output on file OUTPUT.
B=lfn$_5$	Use file lfn$_5$ for the replacement file.
B=0	Do not use a default replacement file.
B omitted	Use file LGO as the default replacement file.

C — Copy the new program library file over the old program library file after processing.

C omitted — Do not copy the new program library file over the old program library file after processing.

R — Do not rewind program library files after processing.

R omitted — Rewind old and new program library files after LIBEDIT and VFYLIB processing.

V — Call VFYLIB after LIBEDIT processing.

V omitted — Do not call VFYLIB to verify program libraries after LIBEDIT processing.

D — Ignore errors and continue.

D omitted — Do not ignore errors; abort job.

For a description of the LIBEDIT directives and examples of their use, refer to appendix C.

LIBGEN STATEMENT

The LIBGEN control statement allows the user to generate a user library file.

The control statement format is:

```
LIBGEN(p1,p2,...,pn)
```

p_i — Any of the following in any order:

F=lfn_1 — Name of source file containing records to be placed on user library file lfn_2.

F — System assumes source file LGO.

F omitted — System assumes source file LGO.

P=lfn_2 — Name of the file on which the user library is to be written.

P — System assumes user library to be written on ULIB.

P omitted — System assumes user library to be written on ULIB.

N=lfn_3 — Name of the user library being generated; this name becomes the name of the ULIB and OPLD records.

N — System assumes lfn_3=lfn_2.

N omitted — System assumes lfn_3=lfn_2.

NX=n — If n is nonzero, no cross-references are given. That is, decks are not cross-linked in the ULIB directory. This can be used to avoid duplicate entry points on loads.

NX omitted — The system assumes n=0.

LIBGEN processes the source file specified and generates a user library file on the file specified with the P parameter. The user library is given the name specified with the N parameter. If the F and P options specify the same file, the message

```
FILE NAME CONFLICT.
```

is issued.

The F and P parameters may appear more than once. In such a case, the last occurrence is used.

LIBGEN rewinds and scans the source file and builds a directory of all entry points, program names, and external references for records in the file. When an EOF mark appears, LIBGEN terminates the directory and rewinds lfn_1. LIBGEN then copies lfn_1 to lfn_2, adding the library and directory records. The directory is written as the first record of the new file. It is indicated as a user library type record by a 76 identification table. The identification table also contains the name of the library.

The directory contains all external references within the library and the linkage to routines that reside in the library. This indicates which routines must be loaded when routines from this library are loaded. This means that all externals for routines in a user library are automatically satisfied from that library first.

The entire file follows the directory record on the new file. The file index is the last record on the file. This record contains random addresses for each record in the file. The index record has a table identifier of 7000_8. LIBGEN processes REL type records, bypassing all other record types.

For example, file RELB contains routines that are used at execution time for several application programs. It is desirable to load these routines as needed when executing the application programs. To generate the user library, the following control statement

```
LIBGEN(F=RELB, P=MYLIB, N=APPLIB)
```

is entered. This creates user library APPLIB on file MYLIB. If FORTRAN application programs are compiled using the control statement.

```
FTN.
```

the user library can be used by loading the program in the following manner.

```
LDSET(LIB=MYLIB/RUNLIB)
LOAD(LGO)
EXECUTE.
```

This causes the program to be loaded and executed with externals satisfied first from user library MYLIB, then from user library RUNLIB, and finally from the system default library SYSLIB.

For examples of the use of LIBGEN, refer to appendix C.

LIST80 STATEMENT

The LIST80 routine reads a file containing COMPASS source code 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, . . . , 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

N1+N2+N3+N4+N5+N6 must be less than or equal to the number of columns specified in the H parameter.

Ix=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 table 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.? 9Ø (CR)
   NO. OF  MOVED FROM  MOVED TO
   CHARS.    COLUMN     COLUMN
(X) (NX)      (IX)       (OX)
 1.  72        1          1
 2.  Ø         Ø          Ø
 3.  Ø         Ø          Ø
 4.  Ø         Ø          Ø
 5.  Ø         Ø          Ø
 6.  Ø         Ø          Ø
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, current working files 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.

Note that all current working files 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 is used to release 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.

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 allows the user to pack a specified file and copy it to another. 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.

The programmer should note that problems may arise when using PACK with direct access files. For example, if file A resides on a legal direct access file device and the following cards are submitted:

```
PACK(A)
DEFINE(A)
```

PACK may copy file A to a device which does not support direct access files. In this event, the DEFINE statement would then cause the job to abort and the following message to be issued to the user's dayfile

```
DIRECT ACCESS DEVICE ERROR,AT nnn.
```

where nnn is the file environment table (FET) address. †

The user can avoid this situation by defining file A as an empty direct access file, creating the file, and then packing it.

```
DEFINE(A)
create file A
PACK(A)
```

The following error messages may be issued to the user's dayfile in response to a PACK statement.

Message	Description
PACK PARAMETER ERROR.	The PACK control statement contains an error.
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 represents data typed at the terminal keyboard, and therefore, cannot be packed).
ILLEGAL CIO REQUEST.	An attempt was made to pack a nonmass storage file.
WRITE ON READ-ONLY FILE fff, AT nnn.	The direct access file was not attached in write mode (refer to ATTACH Statement, section 8).

† Refer to Permanent File Manager, section 5, volume 2.

PRIMARY STATEMENT

The PRIMARY control statement makes a local file the primary file.

The control statement format is:

PRIMARY(lfn)

lfn	Name of local file

The file to be made primary must be a local mass storage file. 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 is used to change 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 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 a print type file.

REQUEST STATEMENT

The REQUEST control statement enables the user to assign a file to a device by including in the comment field a description of an acceptable device.

The control statement format is:

REQUEST(lfn, $\left\{\begin{matrix}\text{CK}\\\text{CB}\end{matrix}\right\}$)

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.

The descriptive comment is displayed at the system console, directing the operator to make the requested assignment.

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.

† Refer to LIMITS control statement, section 6.

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.
```

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 à 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, section 10.

† Any mass storage file used as a checkpoint file must have write permission.

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 file to be resequenced

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.

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 attached to the beginning of each line. If the line number encountered or required exceeds 99999, the following message is issued.

```
LINE NUMBER LIMIT EXCEEDED.
```

Some BASIC statements reference the sequence numbers which must also be changed; therefore, it is imperative that the user specify the proper file type (t). When errors occur while resequencing a BASIC program, the following message is issued for all lines containing errors.

```
ERROR AT LINE xxx.
```

The file being resequenced by the RESEQ statement must have previosuly been sorted. Results are unpredictable if this requirement is not met.

RETURN STATEMENT

The RETURN control statement releases the specified file from job attachment and/or releases its file space.

The control statement format is:

RETURN(lfn_1, lfn_2, . . . , lfn_n)

lfn_i — Name(s) of the file(s) to be returned

The operation performed depends on the file type.

Type	Operation
Input	The file name is changed to INPUT*. File space is not released; INPUT* remains attached to the job as a local file (refer to Input File Control in section 3 for further information).
Print	Job attachment and file space are released.
Punch	Job attachment and file space are released.
Local	Job attachment and file space are released.
System	Job attachment is released but file space remains.
Library	Job attachment is released but file space remains.
Primary	Job attachment and file space are released.
Permanent	Write interlock is cleared. Job attachment is released but file space remains.

In addition, the RETURN statement can be used to decrease the number of tapes or packs scheduled for the job via the RESOURC control statement. However, the number of tapes or packs scheduled is decremented only if the number of tapes or packs scheduled have actually been assigned to the user's job.

REWIND STATEMENT

The REWIND control statement causes files to be rewound and positioned to the BOI (or beginning-of-reel for magnetic tape files).

The control statement format is:

REWIND(lfn_1, lfn_2, . . . , lfn_n)

lfn_i Name(s) of file(s) to be rewound

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 X, S, L, E, B, 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 Tape Management control statements 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 releases or prepares for release a designated file to the input or output queue. The parameters associated with the file may take effect when the statement is processed or may be deferred to a later job step or job termination. This statement also allows the user to rescind a prior deferred ROUTE statement, thereby making the named file type LOFT.

The control statement format is:

ROUTE(lfn,p_1,p_2,...,p_n)

lfn — Name of the file to route; depending on the options specified, the file may be created or it may preexist.

The remaining parameters are order-independent, but at least one must appear.

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.)
LP	Print on any printer
PR	Same as LP
P2	Print on 512 printer
LQ	Same as P2
LR	Print on 580-12 printer
LS	Print on 580-16 printer
LT	Print on 580-20 printer
SB	Punch system binary
PB	Same as SB
P8	Punch 80-column binary
PU	Punch coded
PH	Same as PU
SC	Rescind prior routing and make file LOFT

If the DC parameter is omitted, the default will depend on whether or not a special file name is specified for lfn. If lfn is not a special file name, the default is DC=SC. If lfn is a special file name, DC will assume one of the following values.

Special File Name	Associated DC
OUTPUT	DC=LP
PUNCH	DC=PU
PUNCHB	DC=SB
P8	DC=P8

p_i Description

DEF Indicates that routing of the file to the queue will be deferred to a later job step or end of job. If this parameter is specified, the file will be created if it does not exist. DEF is not allowed if DC=IN.

EC=xx Defines external characteristics for print- or punch-type files.

For print-type files, xx may specify the following values.

A4	ASCII 48-character set
A6	ASCII 64-character set
B4	Display code 48-character set
B6	Display code 63/64-character set

For punch-type files, xx may specify the following values.

ASCII	Punch ASCII
O26	Punch O26 mode
O29	Punch O29 mode
SB	Punch system binary
80COL	Punch 80-column binary

NOTE

If the user includes the EC parameter on a ROUTE statement, the file queue processor may be unable to select that file for output. If EC is not specified, an appropriate EC default is established on the basis of the disposition code (DC) and installation options. Accordingly, the EC parameter is not normally specified. However, if the user does include this specification, the xx selected must be consistent with the queue file processor (BATCHIO, EI200, RBF, etc.).

FC=xx Forms code; specifies that a special form must be placed in the output device before the named file will be selected from the queue. xx can be any two alphanumeric characters, but the combinations null, AA, AB, AC, AD, AD, AE, and AF will give maximum system efficiency. A value of null results when no FC parameter is specified.

FID=xx This is an NOS/BE parameter included for compatibility. It produces an informative message under NOS.

FM Implicit remote routing (refer to the following note).

FM=xx Family name; indicates routing to a remote terminal driven by EI200 or RBF. Normal default procedures apply if this parameter is not specified.

p_i Description

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 similar to the ID specified formally by the SETID control statement.)

ID Implicit central site routing (refer to the following note).

PRI=xx File priority. This is an NOS/BE parameter included for compatibility. It produces an informative message under NOS.

REP=xx Specifies a file repeat count from 0 to 31 (decimal default). Values beyond this range are set to zero (default), and an informative message is issued. The value zero is handled internally to produce one listing.

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 an NOS/BE parameter included for compatibility. It produces an informative message under NOS.

TID Implicit remote routing (refer to the following note).

TID=C Central site routing. This is an 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 following note).

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 following note).

NOTE

For jobs of EIOT origin, the following action is taken.

- Parameter ID, ID=xx, or TID=C will cause routing to the central site.
- Parameter FM, TID, or UN with no argument will cause routing to the terminal of origin.
- The omission of FM, TID, or UN will cause routing to the terminal of origin.
- Parameter FM or UN with legal arguments will cause 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 will cause routing to the central site.
- Specifying UN, TID, or FM without parameters will cause 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 will cause 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 will be 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 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.

The system does not backspace past the beginning-of-information (BOI), in the event that BOI 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; $0 \le \ell \le 17$. If $0 \le \ell \le 16$, 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.

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 the file is to be sorted on; $n \le 10$. 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.

The following SORT error messages may be issued to the user's dayfile.

Message	Description
NO LINE NUMBER ON SORT FILE.	A line on the input file is missing a line number or a line exceeded the 150-character limit.
ILLEGAL SORT PARAMETER.	The SORT control statement is in error.
EMPTY SORT INPUT FILE.	File lfn contains no data.
WRITE ON READ-ONLY FILE fff AT nnn.	The direct access input file was not attached in write mode (refer to ATTACH Statement, section 8).

STAGE STATEMENT

The STAGE control statement causes files to be copied from the specified device to a file residing on mass storage.

The control statement format is:

STAGE(lfn, p_1, p_2, . . . , p_n)

lfn — Name associated with file to be staged from magnetic tape to mass storage

p_i — Any of the following in any order:

NR — Do not rewind lfn before beginning operation; default is rewind.

NU — Do not unload lfn after staging operation; default is automatic unload.

DR — Drop job after staging operation.

N=n — Copy n files to lfn.

T=xx — Stage file lfn from device with EST ordinal xx. † This parameter is specified only when tape containing files to be staged is unlabeled (X format and system default density).

VSN=vsn — Specifies the 1- to 6-character volume serial number of the labeled tape containing the file to be staged

D=den — Tape density:

200	200 bpi (implies 7-track)
556	556 bpi (implies 7-track)
800	800 bpi/cpi (7- or 9-track)
1600	1600 cpi (implies 9-track)

F=format — Data format (refer to section 10):

I	Internal
X	External
SI	System Internal††

MT — 7-track tape (default)

NT — 9-track tape

If T is not included but VSN is included, n files are copied from the specified tape. If neither T nor VSN is included, a request for lfn is issued to the operator. If DR is not included, STAGE requests the next set of parameters for the next staging operation to be entered by the K display on the system console. When lfn is staged to mass storage, it is designated as a library file. If a library file already exists with the same name as the file being staged, the system issues the following message.

```
DUPLICATE NAME.
```

† Contact installation personnel for a list of EST ordinals.

†† NOS/BE system default tape format.

TDUMP STATEMENT

The TDUMP control statement lists a file in octal and/or alphanumeric form.

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	Input file name (default is TAPE1)
L=lfn_2	Output file name (default is OUTPUT)
O	Octal dump only (default is octal and alphanumeric dump)
A	Alphanumeric dump only (default is octal and alphanumeric dump)
R=rcount	Number of records in decimal to dump (default is dump to EOI)
F=fcount	Number of files in decimal to dump (default is dump to EOI). If F=0, dump continues until an empty file (double EOF) is encountered.
N=lines	Maximum number of lines in decimal that can be dumped (if N is omitted, there is no restriction on the number of lines).
NR	Do not rewind file lfn_1 before dump (default is to rewind lfn_1).

The user has the option of dumping the entire file or of specifying the number of records, files, or lines to dump.

UNLOAD STATEMENT

The UNLOAD control statement releases job attachment and/or the file space of the specified file.

The control statement format is:

UNLOAD(lfn_1, lfn_2, . . . , lfn_n)

lfn_i Name(s) of the file(s) to be unloaded

The UNLOAD statement performs the same function as the RETURN control statement (for additional information, refer to the description of the RETURN statement earlier in this section). Unlike the RETURN statement, an UNLOAD of a magnetic tape file cannot be used to decrease the number of tape units scheduled for the job via 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.

1. If the tape is ANSI labeled, the system writes a tape mark, an EOF1 label, and three tape marks and then unloads the tape.

2. If the tape is unlabeled and the data format specified on the ASSIGN, LABEL, or REQUEST card is X, S, L, E, B, or F, the system writes four tape marks and then unloads 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 unloads the tape.

Refer to Tape Management control statements, 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, . . . , 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 TYPE.

Common 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 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, the system assumes TAPE1.

lfn_2 Name of the second file; if this parameter is omitted, the system assumes TAPE2.

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 either file.
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 if errors occur.
R	Rewind both files before and after the verify.
C	Use coded file mode for SI, S, and L format coded tapes.

Whenever words on the two files do not match, VERIFY lists the:

- Record number
- Word number within the record
- Words from both files that do not match

If errors are encountered, the following 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 message is issued.

FILE NAME CONFLICT.

VFYLIB STATEMENT

The VFYLIB control statement performs a binary comparison of two specified files after rewinding both files.

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, the system assumes OLD.

lfn_2 Name of the second file; if this parameter is omitted, the system assumes NEW.

lfn_3 Name of the file to receive output; if this parameter is omitted, the system assumes OUTPUT.

NR If specified, lfn_1 and lfn_2 are not rewound.

The VFYLIB program lists

- Replacements
- Deletions
- Insertions

on the output file lfn_3. A program is defined as being replaced when the actual binary code is changed. Information in the prefix (77) table such as last modification date and last assembly date is skipped in VFYLIB's comparison.

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:

WRITEF(lfn, x)

lfn Name of the file to be written on

x Number of file marks to be written; if this parameter is omitted, the system assumes x=1.

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
ATTACH
CATLIST
CHANGE
DEFINE
GET
OLD
PACKNAM
PERMIT
PURGALL
PURGE
REPLACE
SAVE

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. 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 is unable to 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.
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 or defined. 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.

†CT=LI can also be used to specify public files.

Keyword	Option	Description	
M=	m	File or user permission mode:	
		W or WRITE	Allows the user to write, read, append, execute, modify, and/or purge the file. This mode can be specified for direct or indirect access files.
		M or MODIFY	Allows the user to modify, append, read, and/or execute a direct access file. Adding new information within the existing boundaries of the file is legal but the file size must be maintained.
		A or APPEND	Allows the user to append information to the end (EOI) of the file. This mode can be specified for direct or indirect access files.
		R or READ	Allows the user to read and/or execute the file. This mode can be specified for direct or indirect access files.
		RM or READMD	Allows the user to read and/or execute a direct access file with the implication that another user may currently be accessing the same file in MODIFY mode. This mode can be specified only for direct access files.
		RA or READAP	Allows the user to read and/or execute a direct access file with the implication that another user may currently be accessing the same file in APPEND mode. This mode can be specified only for direct access files.
		E or EXECUTE	Allows the user to execute the file. If the file is attached to the user's job in EXECUTE mode, the file must be in absolute format. This mode can be specified for direct or indirect access files. Relocatable files with EXECUTE permission may be loaded and executed only via a stand-alone file name call (such as LGO) which is not preceded by a loader control statement.
		N or NULL	Removes permission previously granted via PERMIT control statements. This mode can be specified for direct or indirect access files.

Keyword Option Description

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≤i≤8)
DJi	844-41/44 Disk Storage Subsystem (1≤i≤8)
DP	Distributive Data Path to ECS†
MDi	841 Multiple Disk Drive (1≤i≤8)

The R keyword can be used in two ways.

1. It can be used on the DEFINE control statement to specify the family device on which the direct access permanent file is to reside.
2. 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.

S= space

Specifies the amount of space in decimal PRUs desired for the direct access file. Refer to the DEFINE control statement.

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.

† The job must be of system origin or the user must be validated for system origin privileges.

Keyword	Option	Description
NA		The NA keyword can be used in two ways. 1. 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. 2. 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 user's working files 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$_i$.

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.

The user should return a file as soon as possible since this usually increases the availability of the file to other alternate users.

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)
```

p One of the following list options:

F Selects a listing of pertinent information about each file in the user's catalog. 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. Note that 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 Selects a listing of 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).†

0 (zero) Selects a short list that includes only the names of the 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.

P Selects a short list that indicates only the user numbers of alternate users who have accessed the specified private or semiprivate file. This option requires that a file name be specified (FN option).

† User numbers are not recorded for accesses to public files.

pfn

Permanent file name. This option specifies that catalog information is desired only for this permanent file. This parameter is required when listing permit information (LO=FP, LO=P). If the short list options are selected (LO=0, 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 specified file (or user number) is not located.

usernum

User number. This parameter has two purposes.

1. For LO=F and LO=0. Indicates the alternate catalog for which the user desires catalog information.
2. For LO=FP and LO=P. Indicates the permission information recorded for the specified alternate user.

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.

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 — 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.

CHANGE STATEMENT

The CHANGE control statement allows the originator of a direct or indirect access file to alter any of several parameters without having to attach and redefine the file or retrieve and save it.

The control statement format is:

```
CHANGE(nfn=ofn/CT=ct, M=m, PW=passwrd, PN=packname, R=r, NA)
```

nfn — New permanent file name

ofn — Old permanent file name. If no name change is desired, only ofn is specified.

The CT, M, and PW keywords should be specified only if a change in the value associated with that keyword is desired. To clear the password for an existing file, the user must set PW=0. The PN and R keywords cannot be used to specify a new auxiliary device. They are used only to specify the device on which ofn resides. CHANGE also updates the last modification date and last access date for the specified file.

The following messages may be issued to the user's dayfile in response to a CHANGE request.

Message	Description
ofn NOT FOUND, AT nnn.	The specified permanent file, ofn, was not found in the user's catalog.
nfn ALREADY PERMANENT, AT nnn.	The new permanent file, nfn, already exists in the user's permanent file catalog.

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)

lfn_i If DEFINE is to be used to create 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 is to be used to define 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.

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.

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.

If the request is for a file in another user's catalog (UN option specified), the permission mode is that which the user has been permitted for private files or that specified in the catalog for semiprivate and public files.

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 working files are also 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, section 7 for use of primary file types.

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 Device Residence, 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 card 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 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.

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 allows a user to remove a file 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 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, the user must have write permission or the file must be semiprivate or public with write mode. If pfn$_1$ 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, NA)

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. 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, 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, 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 local files are rewound when the save operation is completed. If the optional parameters are omitted, the system assumes the following values.

Keyword	Default
PW	None
CT	PRIVATE
M	WRITE
PN	None

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.
```

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	LOC	RBR
DMD	PBC	WBR
LBC		

NOTE

For information concerning security restrictions associated with the use of these control statements, refer to Security Control, section 3.

The DMP and DMD control statements dump central memory in octal representation and/or display code equivalences. These statements are particularly helpful in creating dumps for debugging purposes. (Refer to Debugging Aids, section 13.) 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 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, DMP assumes fwa=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.

The DMP routine dumps on file OUTPUT central memory according to the DMP call parameters in four words per line. If lines are duplicated, they are suppressed with the following notation.

DUPLICATE LINES.

The DMP statement must immediately follow the program to be dumped, except that another DMP, DMD, or EXIT statement may intervene.

Dumping will always stop at FL if lwa > FL. If either fwa or lwa is nonnumeric, the request is interpreted as:

DMP.

If fwa ≥ FL, fwa is set to FL-10_8. If both fwa and lwa > FL, fwa is set to FL-10_8 and lwa is set to FL. If fwa=lwa, the system adds 10_8 to lwa and proceeds with the operation. If fwa ≥ 400000_8, the first dump address is fwa-400000_8, memory from the first dump address through lwa is dumped, and the job is aborted. If fwa ≥ lwa, the system issues the following message to the user's dayfile.

DUMP FWA .GE. LWA+1.

Since the user's FL is not saved between commands entered from a time-sharing terminal, the only way to use DMP from a terminal is to call a procedure file. A dump from a terminal is formatted for 72-column output.

DMD STATEMENT

The DMD control statement requests a dump similar to that of the DMP statement but additionally contains 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 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=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 is not allowed from a time-sharing terminal.

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).

The following messages may be issued to the user's dayfile in response to an LBC card.

LBC ARGUMENT ERROR.	The load address, addr, is nonnumeric.
LBC FWA .GE. FL.	The load address is greater than or equal to the user's field length.
RECORD TOO LONG.	The record is too long for available memory. Available memory is filled and the excess data is skipped.

LOC STATEMENT

The LOC control statement calls the LOC program and specifies address parameters used by LOC to read octal line images from file INPUT and enter them in CM.

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.

The correction statement images consist of octal address and data fields. 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 five characters or less, it is separated from the data field by a nonoctal character (for example, a blank). If it is six characters, no separator is required.

The data field consists of 1 to 20 octal chracters. 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

The following messages may be issued to the user's dayfile.

LOC ARGUMENT ERROR.	Either fwa or lwa is nonnumeric.
LOC RANGE ERROR.	Either fwa is greater than lwa or lwa is greater than FL.
ADDRESS OUT OF RANGE, aaaaaa.	The address aaaaaa on a correction statement is greater than or equal to the user's field length. The correction statement is ignored and LOC continues.

If both addresses are specified and both are nonzero, storage is cleared from fwa to lwa and the octal line images are loaded at the specified addresses. LOC can be called to clear storage by providing an empty (zero-filled) record on file INPUT.

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=RA. If lwa=fwa, and a nonzero value is specified, PBC adds 10_8 to lwa. If fwa and lwa=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.

The following messages may be issued to the user's dayfile.

PBC ARGUMENT ERROR.	Either fwa or lwa is nonnumeric.
PBC FWA .GT. LWA.	The fwa parameter is greater than lwa.
PBC RANGE ERROR.	The lwa parameter is greater than or equal to the user's field length.

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 four characters and is numeric, TAPEn is the file name. If n contains a nonnumeric character or is four 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.

	59	53	47	35	17 0
RA	77	00	0016	0	length
RA+1	name				0
RA+2	date (yy/mm/dd.)				
RA+3	0				
•					
•					
RA+17	0				
RA+20_8	5200		0		length_1

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 four characters and is numeric, TAPEn is the file name. If n contains a nonnumeric character or is four 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.

The following messages may be issued to the user's dayfile.

WBR ARGUMENT ERROR.	The rl parameter is nonnumeric.
RECORD TOO LONG.	The rl parameter is greater than or equal to the user's field length.

TAPE MANAGEMENT 10

This section is devoted primarily to the control statements necessary to create and manage files on magnetic tape. Following the control statements are descriptions of the various types of tape formats available to the user.

The control statements described in this section are:

ASSIGN
BLANK
LABEL
LISTLB
REQUEST
VSN

The ASSIGN, LABEL, and REQUEST control statements cause files to be assigned to tape units or devices. The REQUEST statement requires operator action unless the VSN is specified. In this case, if the tape has already been mounted, assignment is automatic. LABEL and ASSIGN also cause automatic assignment.

ANSI tape labels can be read using the LISTLB statement and blank-labeled for installation control using the BLANK statement.

The control statements available to the user for assigning a file to magnetic tape are also used to create new and access existing 7- and 9-track labeled and unlabeled tapes. The following terms are used in describing these statements.

Volume — Reel of magnetic tape

Volume serial number — Number that uniquely identifies a reel of tape

Block — One physical record unit (PRU); that is, a group of contiguous characters recorded on and read from magnetic tape as a unit.

Noise — Any block less than the minimum acceptable block size is considered noise and discarded by the system.

Label — Field of characters that identifies and/or delimits a volume or file. Labels may be written in ANSI standard or nonstandard format. ANSI labels are 80-character blocks recorded at the beginning of a volume (VOL1), the beginning of a file (HDR1), the end of a file (EOF1), and the end of a volume (EOV1). Labels which do not conform to ANSI standards in format and/or content are defined as nonstandard.

Tape mark — Special configuration recorded on magnetic tape indicating the boundary between files and/or labels

Owner — Owner of a NOS written tape identified in the VOL1 label by the combination of his family name and user number

NOS written tape — Tape with ANSI labels written by NOS and identified as such in the system code field of each HDR1, EOF1, and EOV1 label

The format and contents of ANSI labels are described in appendix G. A RESOURC control statement must be included in any job that uses two or more tape units concurrently.

The following is a list of the parameters that may appear on one or more of the tape management control statements.

Keyword	Parameter	Default	Valid On	Description
C=	ccount	None	ASSIGN REQUEST	Character count. Specifies the maximum size block (in 6-bit characters) that may be read or written. This parameter applies only to E, B, and F data formats (refer to the F keyword).
CB		Refer to REQUEST statement.	ASSIGN LABEL REQUEST	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.
CK		Refer to REQUEST statement.	ASSIGN LABEL REQUEST	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.
CR= or C=	cdate	Current date	LABEL	Creation date in the form yyddd where 1≤ ddd≤ 366. Creation date is meaningful only on read operations; on write operations, the current date is always used (refer to appendix G, HDR1 label).
CV= or N=	conv	Installation parameter	ASSIGN BLANK LABEL REQUEST	Specifies the conversion† mode for 9-track tapes: AS ASCII/display code conversion US Same as for AS EB EBCDIC/display code conversion For unlabeled 9-track I or SI (internal) format tapes, conversion is always forced to ASCII.

† Refer to ASCII/Display Code and EBCDIC/Display Code Conversion, appendix A.

Keyword	Parameter	Default	Valid On	Description
D=	den	Installation parameter	ASSIGN BLANK LABEL REQUEST	Specifies tape density: LO 200 bpi (7-track) HI 556 bpi (7-track) HY 800 bpi (7-track) HD 800 cpi (9-track) PE 1600 cpi (9-track) 200 200 bpi (7-track) 556 556 bpi (7-track) 800 800 bpi (7- or 9-track) 1600 1600 cpi (9-track)
E=	gvn	Refer to appendix G.	LABEL	1- to 2-digit generation version number (refer to appendix G, HDR1 label).
F=	format	I format for LABEL statement. I format for ASSIGN statement if VSN keyword is included; otherwise, default is X format. I format for REQUEST statement if VSN control statement has been processed; otherwise,default is X format.	ASSIGN LABEL REQUEST	Specifies the data format I Internal X External B Blocked E Line image S Stranger L Long block stranger SI System internal[†] F Foreign
FA=	fa	Unlimited access	BLANK LABEL	File accessibility. One character that indicates who has access to the labeled file (refer to appendix G, HDR1 label). A Only the owner of the NOS written tape can access the file. omitted FA omitted indicates unlimited access. other All future accesses to this tape must specify this character as the fa parameter. File accessibility is not checked for system origin jobs.
FC=	fcount	None	ASSIGN LABEL REQUEST	Frame count. Specifies the maximum size block (in frames) that may be read or written. This parameter applies only to E, B, and F data formats (refer to the F keyword).

[†] NOS/BE system default tape format.

Keyword	Parameter	Default	Valid On	Description
FI= or L=	fileid	Refer to appendix G.	LABEL	A 1- to 17-character file identifier recorded in HDR1.
G=	genno	Refer to appendix G.	LABEL	1- to 4-digit generation number (refer to appendix G, HDR1 label).
HD		Installation parameter	ASSIGN REQUEST	800-cpi tape density (implies 9-track tape)
HI		Installation parameter	ASSIGN REQUEST	556-bpi tape density (implies 7-track tape)
HY		Installation parameter	ASSIGN REQUEST	800-bpi tape density (implies 7-track tape)
L=	out	OUTPUT	LISTLB	Specifies the file on which the labels are to be listed.
LB=	ℓ	KL for LABEL statement. KL for ASSIGN statement if VSN keyword is included; otherwise, default is KU. KL for REQUEST statement if VSN keyword is included or VSN control statement has been processed; otherwise, default is KU.	ASSIGN LABEL REQUEST	Specifies whether the tape is to be treated as labeled or unlabeled. KU NOS unlabeled. KL NOS labeled. The tape is treated as having ANSI labels. If the tape is a NOS tape, volume and header accessibility restrictions will be enforced. NS Nonstandard labels. The system skips over labels based on tape marks but does not process the labels. This option can also be used in processing a non-NOS tape that, although designated as unlabeled, contains a tape mark prior to the beginning of data.
LO		Installation parameter	ASSIGN REQUEST	200-bpi tape density (implies 7-track tape)
LO=	ltype	R	LISTLB	Specifies the type of labels to be listed (refer to appendix G). A List all required and optional ANSI labels.

Keyword	Parameter	Default	Valid On	Description
				R List all required labels (VOL1, HDR1, EOF1, and if present, EOV1)
				O List all optional labels (VOL2-9, HDR2-9, EOF2-9, EOV2-9, UVL1-9, UHLa, and UTLa)
				V List all VOL1-9 labels.
				H List all HDR1-9 labels.
				F List all EOF1-9 labels.
				E List all EOV1-9 labels.
				U List all UVL1-9, UHLa, and UTLa labels.
LSL=	lsl	The labels and data format of the specified volume require the agreement of the interchange parties.	BLANK	Label standard level (refer to appendix G, VOL1 label). 1 The labels and data format of this volume conform to the requirements of the ANSI standard. blank LSL omitted indicates that the labels and data format of the specified volume require the agreement of the interchange parties.
MT		Installation parameter	ASSIGN BLANK LABEL REQUEST	Specified file resides on or is to reside on 7-track tape †
NT		Installation parameter	ASSIGN BLANK LABEL REQUEST	Specified file resides on or is to reside on 9-track tape †
NS=	ns	For all data formats except I, SI, and X, the default for ns is 18 frames. The NS keyword should not be specified for I, SI, and X format tapes because the definition of noise block size is implied by the format. (Refer to the F keyword.)	ASSIGN LABEL REQUEST	Noise size. Any block containing fewer than ns frames is considered noise and discarded by the system; the maximum value of ns is 31 frames. If a noise size of zero is specified, the default noise size is used.

† If MT or NT is used and disagrees with the track type implied by the density selected, the MT/NT CONFLICT message will be issued.

Keyword	Parameter	Default	Valid On	Description
OFA=	fa	Unlimited access	BLANK	One character that indicates the current file accessibility of a labeled tape which is to be blank labeled. (Refer to FA description for explanation of fa characters.)
OWNER=	usernum/ familyname	Refer to appendix G.	BLANK	Identifies the owner of the NOS labeled tape (refer to appendix B, VOL1 label).
PE		Installation parameter	ASSIGN REQUEST	1600-cpi tape density (implies 9-track tape)
PO=	$p_1p_2\ldots p_n$	U for system origin jobs	ASSIGN LABEL REQUEST	A string of characters specifying processing options: A Job will be automatically aborted on an irrecoverable read or write parity error regardless of ep bit. N Job will not be aborted if an irrecoverable read or write parity error occurs regardless of ep bit; on a read operation, data will be passed to the program. R Enforce ring out. If the tape is mounted with the write ring in, job processing will be suspended until the operator remounts the tape correctly. W Enforce ring in. If the tape is mounted without the write ring in, job processing will be suspended until the operator remounts the tape correctly. 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 origin type jobs, omission of the U option causes the tape to be unloaded at end of usage.

Keyword	Parameter	Default	Valid On	Description
				F Force unload. Unload at the end of usage. This option is useful for system origin jobs where otherwise U (inhibit unload) would be the default.
				E Error inhibit. All hardware read/write errors are ignored and processing continues. The system does not attempt error recovery, issue error messages, nor return error status. This option is not intended for the normal user. However, it can be used to recover portions of data from a bad tape, for hardware checkout purposes, and to write on tape without skipping bad spots; in the latter case, the user is responsible for verifying that the data was written correctly.
				B Directs the system to write system noise blocks when performing write error recovery. This option is ignored for 1600-bpi tapes. In addition, this option should not be used for tapes which are to be interchanged with other systems.
				I Directs the system to ignore the block being read when the EOT is encountered. †
				P Directs the system to accept the block being read when the EOT is encountered. †

† For further information, refer to End-Of-Tape/End-Of-Reel Conditions at the end of this section.

Keyword	Parameter	Default	Valid On	Description
				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.
QN= or P=	seqno	Refer to appendix G.	LABEL LISTLB	1- to 4-digit file sequence number (refer to appendix G, HDR1 label). [†]
R		R	LABEL	Directs the system to read the existing ANSI label. The parameters on the LABEL statement are compared with the values recorded on the file labels. If the comparison fails, the job is aborted.
RT=	rd	Current date	LABEL	Retention date in the form yyddd (used to derive expiration date described in appendix G, HDR1 label).
SI= or M=	setid	Refer to appendix G.	LABEL LISTLB	1- to 6-character set identifier for a multifile set (refer to appendix G, HDR1 label). [†]
SN= or V=	secno	Refer to appendix G.	LABEL	1- to 4-digit file section number (refer to appendix G, HDR1 label).
T=	retcycle	Refer to appendix G.	LABEL	1- to 3-digit retention cycle specifying the number of days from the current data that the file is to be retained (used to derive the expiration date described in appendix G, HDR1 label, if the RT keyword is not specified).

[†] Refer to LABEL Statement in this section for constraints on using the QN and SI parameters.

Keyword	Parameter	Default	Valid On	Description
U		Inhibit physical unload of tape at end of usage.	BLANK	Clears inhibit unload (PO=U) processing option; thus, tape is physically unloaded when returned after blank labeling. This does not apply to system origin jobs.
VA=	va	Unlimited access	BLANK	Volume accessibility. One character that indicates restrictions on who may have access to information on the reel (refer to appendix G, VOL1 label). omitted VA omitted indicates unlimited access. other Whenever this reel is processed under NOS as an NOS labeled tape (LB=KL), volume accessibility restrictions are imposed. Thus, the user cannot change or destroy the VOL1 label on the tape. This feature enables an installation to blank label new tapes and be assured that the volume serial number field of VOL1 cannot be changed by a user. If VA is nonblank, only a system origin job can change VOL1.
W		R	LABEL	Directs the system to write standard ANSI labels using the parameters specified on the LABEL statement or their default values. It is not necessary to specify the PO=W option to enforce ring in. If the tape is mounted without the write ring, job processing is suspended until the operator remounts the tape correctly. However, if the PO=R option is specified, the job is aborted.
VSN=	vsn	Refer to appendix G.	ASSIGN BLANK LABEL REQUEST	A 1- to 6-character volume serial number that uniquely identifies a reel of tape (refer to the VSN control statement).

The system allows continuation lines for ASSIGN, BLANK, LABEL, REQUEST, and VSN statements that require 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. All continuation lines must contain a blank in column 1.

NOTE

> The system accepts continuation lines from a time-sharing terminal only if they are contained in procedure files.

The programmer can use a literal for any parameter on a tape management control statement that contains nonalphanumeric characters. Characters other than letters, numbers, and asterisks are defined as nonalphanumeric. 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
```

If continuation cards are used, a literal cannot extend from one card to another.

Generally, if more than one parameter of a given type is specified, the last one encountered in a left-to-right scan is used. The two exceptions to this rule are in the processing option parameters. If both ring enforcement options (PO=R and PO=W) or more than one EOT option (PO=I, PO=P, PO=S) is specified, the ARGUMENT ERROR message is issued to the user's dayfile.

ASSIGN STATEMENT

The ASSIGN control statement can be used to create a new unlabeled tape or access an existing labeled or unlabeled tape. The following description applies only to magnetic tape devices; for use of the ASSIGN statement with devices other than magnetic tape, refer to section 7.

The control statement format is:

```
                         { FC=fcount }            { MT }
ASSIGN(nn, lfn, D=den,   {  C-ccount }, CV=conv,  { NT }, PO=p1p2...pn,
                                      { CK }
     F=format, NS=ns, LB=1, VSN=vsn,  { CB } )
```

nn Device or device type to which the specified file is to be assigned; nn may be either the EST ordinal[†] of a magnetic tape unit or one of the device types MT or NT. MT is defined as a 7-track magnetic tape drive; NT is a 9-track magnetic tape drive.

lfn Name of the file to be assigned to the specified equipment.

Although the user can also include this statement to assign a labeled tape to his job, he cannot use it to create or verify tape labels. It is suggested that the user include LABEL statements for all tapes whenever possible.

The job must be of system origin or the user must be validated for system origin privileges. The user must also be validated for use of magnetic tapes.[††] If the user attempts to perform an assignment for which he is not validated, the job is aborted and the following message is issued to the user's dayfile.

```
ILLEGAL USER ACCESS.
```

Before performing the assignment, the system issues a RETURN on lfn.

Example:

```
ASSIGN(51, TAPE1)
```

This statement assigns the file TAPE1 to the magnetic tape unit identified by EST ordinal 51.

[†] Contact installation personnel for a list of EST ordinals

[††] Refer to LIMITS control statement, section 6.

BLANK STATEMENT

The control statement format is:

BLANK(D=den, $\left\{\begin{matrix}\text{MT}\\\text{NT}\end{matrix}\right\}$, CV=conv, VSN=vsn, FA=fa, OFA=ofa, VA=va, OWNER=usernum/
familyname, LSL=lsl, U)

With the BLANK control statement, an installation can establish control over the use of labeled tapes. The values supplied on the statement are used to blank label a tape with standard ANSI volume header (VOL1), first file header (HDR1), and first end-of-file (EOF1) labels. The labels are written as follows:

	VOL1	HDR1	*	*	EOF1	*	*		

In writing these labels, the system uses default values for all fields except those fields for which there are corresponding parameters on the BLANK statement. The VA and FA keywords can be used to restrict access to information on the reel and the specified file, respectively. If the tape to be blank labeled is a labeled tape which has a file accessibility other than A, this old file accessibility must be specified by the OFA parameter. When the tape is blank labeled, the file accessibility is that specified by the FA parameter. The default track type may be set by the installation to either MT or NT.† If a track type other than the default is desired, it must be specified.

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 (that is, 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 selecting the write label (W) option.
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 NOS labeled tape (that is, LB=KL), and therefore, cannot change or destroy the VOL1 label.
   - Submit a system origin job to change VOL1.

†Contact installation personnel for the default track type.

LABEL STATEMENT

The control statement format is:

$$\text{LABEL(lfn, D=den, FC=fcount, CV=conv, } \begin{Bmatrix}\text{MT}\\ \text{NT}\end{Bmatrix} \text{, PO=}p_1p_2\ldots p_n\text{, F=format, NS=ns,}$$

$$\text{LB=}\ell\text{ , VSN=vsn, } \begin{Bmatrix}\text{CK}\\ \text{CB}\end{Bmatrix} , \begin{Bmatrix}\text{FI=fileid}\\ \text{L=fileid}\end{Bmatrix} \text{, FA=fa, } \begin{Bmatrix}\text{SI=setid}\\ \text{M=setid}\end{Bmatrix} , \begin{Bmatrix}\text{SN=secno}\\ \text{V=secno}\end{Bmatrix} ,$$

$$\begin{Bmatrix}\text{QN=seqno}\\ \text{P=seqno}\end{Bmatrix} \text{, G=genno, E=gvn, } \begin{Bmatrix}\text{CR=cdate}\\ \text{C=cdate}\end{Bmatrix} , \begin{Bmatrix}\text{RT=rdate}\\ \text{T=retcycle}\end{Bmatrix} , \begin{Bmatrix}\text{W}\\ \text{R}\end{Bmatrix}$$

lfn Name of the file that resides on or is to reside on magnetic tape

The LABEL control statement directs the system to assign file lfn to a tape unit. This assignment occurs using VSN only; the file identifier is not considered in assigning. If a file by the name lfn already exists, the following action is taken.

1. If lfn is assigned to a device other than a tape unit, job processing continues with the next control statement.
2. If lfn is an existing tape file and the read label (R) parameter is specified, the system compares the parameters on the LABEL statement with the values recorded on the file labels. If the comparison fails, the job is aborted.
3. If lfn is an existing tape file and the write label (W) parameter is specified, the system rewrites the header labels (information in HDR1 is not altered). Processing then continues with the next control statement.

To assign to tape an lfn that was previously assigned in the same job to another device, the user must make sure that lfn is returned before the LABEL statement is processed. Note that the default track type may be set by the installation to either MT or NT.† If a track type other than the default is desired, it must be specified. If neither MT nor NT is specified and no VSNs are present, any equipment for which the user is validated may be assigned.

If lfn is to be used for checkpoint dumps and the dumps are to be written on labeled tape, the CK or CB parameter must be included on the LABEL statement. For further information about checkpoint dumps, refer to the REQUEST control statement.

The SI (M) parameter must be present for multifile label positioning using control statements. If the QN (P) parameter is present, the multifile set is positioned to the file set member that matches the specific sequence number. If QN is not specified and the FI (L) parameter is present, the multifile set is positioned to the file set member that matches the file identifier specified. If QN and FI are specified, a match must occur on both sequence number and file identifier. If neither QN nor FI is specified, the tape is positioned to the next file in the multifile set.

To extend a multifile set, QN must be set to 9999.

If the SI parameter is not specified, file positioning is not done. The R and W parameters on the LABEL statement are ignored if SI is specified. If the W parameter is specified, (QN = 1) and it is the first OPEN on the file, an OPEN/WRITE is performed.

† Contact installation personnel for the default track type.

Example 1:

```
        LABEL(NEWFILE,VSN=TP01,FI=FILEA,W)
```

This statement creates an ANSI-labeled tape which the job can access by the filename NEWFILE. Default values are used for all fields of HDR1 except the file identification, FILEA. Any data written is recorded in 512 CM word blocks.

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 OLDFILE. The system compares the vsn in VOL1 and the file identification in HDR1 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, F=I, FI= FIRSTFILE, SI=TEST, QN=1, W)
     COPYBR(INPUT, TAPE)
     LABEL(TAPE, VSN=ONE, F= I, FI=SECONDFILE, SI= TEST, QN=9999)
     COPYBR(INPUT, TAPE, 10)
     RETURN(TAPE)
```

The sequence number QN must equal 9999 to add the second file. This file will be referenced with QN=2 (refer to examples 6, 7, and 8).

Example 4:

The following control statements in a new job add a third file to the multifile set created in example 3.

```
     LABEL(TAPE, VSN=ONE, 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 multifile set created in examples 3 and 4.

```
     LABEL(TAPE, VSN=ONE, F=I)
     LABEL(TAPE, VSN=ONE, F=I, FI=FIRSTFILE)
     LABEL(TAPE, VSN=ONE, F=I, FI=FIRSTFILE, SI=TEST)
```

Positions according to the FI specification. The user will employ this method if the sequential location of the file on the tape is not known.

```
     LABEL(TAPE, VSN=ONE, F=I, QN=1, SI=TEST)
```

Positions according to sequence number.

```
     LABEL(TAPE, VSN=ONE, F=I, QN=1, FI=FIRSTFILE, SI=TEST)
```

Positions by sequence number, but there must be a satisfactory compare of the FI or the job will abort.

Example 6:

Any one of the following control statements can be used to read the second file of the multifile set previously created.

```
LABEL(TAPE, VSN=ONE, F=I, QN=2, SI=TEST)
LABEL(TAPE, VSN=ONE, F=I, FI=SECONDFILE, SI=TEST)
LABEL(TAPE, VSN=ONE, F=I, FI=SECONDFILE, QN=2, SI=TEST)
```

Example 7:

Execution of the following control statements destroys the third file of the multifile set previously created.

```
LABEL(TAPE, VSN=ONE, F=I, QN=2, SI=TEST)
    Positions to the beginning of file 2.
COPYBR(DISK, TAPE)
    Writes a new file 2.
REWIND(TAPE)
    Puts an EOI at the end of file 2.
```

Example 8:

The following example can be used to replace the second file of the multifile set previously created and still retain the first and third files.

```
LABEL(TAPE, VSN=ONE, F=I, QN=3, SI=TEST)
COPYBR(TAPE, DISK, 3)                                         Saves file 3
LABEL(TAPE, VSN=ONE, F=I, QN=2, SI=TEST)
COPYBR(INPUT, TAPE)                                           Replaces file 2
LABEL(TAPE, VSN=ONE, F=I, QN=9999, SI=TEST, FI=THIRDFILE)
REWIND(DISK)
COPYBR(DISK, TAPE, 3)                                         Copies back file 3
etc.
```

LISTLB STATEMENT

The control statement format is:

$$\text{LISTLB (lfn, } \begin{Bmatrix}\text{SI=setid}\\ \text{M=setid}\end{Bmatrix} , \begin{Bmatrix}\text{QN=seqno}\\ \text{P=seqno}\end{Bmatrix} \text{, LO=ltype, L=out)}$$

The LISTLB control statement directs the system to read the ANSI labels on the tape file specified by lfn and write them on the user specified file out. The ltype parameter allows the user to specify the type of labels to be listed (refer to appendix G for a description of each type of label). The setid and seqno parameters are used to list the labels of multifile tapes, as follows:

setid	seqno	Significance
Specified	Not specified	List labels of all files in multifile set.
Specified	Specified	List labels of file with specified seqno only.
Not specified	Specified	Illegal combination; LISTLB aborts.

The user cannot position a multifile tape to a particular file and list the labels for that file. The multifile tape should be positioned at loadpoint, and LISTLB then positions the tape and lists the labels of the desired file. For example, the following lists the labels of file 2 of multifile set ABCDEF.

```
LABEL(T,MT,D=HY,SI=ABCDEF,VSN=EXAMP1)
LISTLB(T,SI=ABCDEF,QN=2)
```

When listing the labels of all files of a multifile set, LISTLB keeps positioning the tape and listing the labels of each file until an end-of-set status is returned to the FET. This causes the following dayfile messages to appear in the user's dayfile

```
MULTI-FILE NOT FOUND, lfn AT 110.
REQUESTED SECTION n+1.
FOUND SECTION n.
```

where n is the last file of the set. These messages also appear if the user requested a file that was not in the set.

To list all the labels of a multireel file, no special parameters or techniques are used. However, to list only the volume and/or header group labels (trailer labels not listed), the user requests each reel separately and employs a LISTLB control statement for each reel. This is necessary since, in this case, no SKIPEI is issued to cause automatic reel switching to take place. Automatic reel switching takes place only if trailer labels are also being listed. For example, the following lists both volume and header group labels of two reels of a multireel file.

```
LABEL(T,MT,D=HY,VSN=REEL1)
LISTLB(T,LO=VH)
RETURN(T)
LABEL(T,MT,D=HY,VSN=REEL2)
LISTLB(T,LO=VH)
```

To list all the labels of a multireel file, only one LISTLB control statement is required. For example:

```
VSN(T=REEL1/REEL2)
LABEL(T,MT,D=HY,VSN=REEL1)
LISTLB(T) or LISTLB(T,LO=R)
```

REQUEST STATEMENT

The REQUEST control statement enables the user to assign a file to a device by including in the comment field a description of an acceptable device.

The control statement format is:

$$\text{REQUEST (lfn, D=den, }\left\{\begin{matrix}\text{FC=fcount}\\ \text{C=ccount}\end{matrix}\right\}\text{, CV=conv, }\left\{\begin{matrix}\text{MT}\\ \text{NT}\end{matrix}\right\}\text{, PO=}p_1, p_2 \ldots p_n\text{, F=format,}$$

$$\text{NS=ns, LB=}\ell\text{, VSN=vsn, }\left\{\begin{matrix}\text{CK}\\ \text{CB}\end{matrix}\right\}\text{)}$$

This comment is displayed at the system console, directing the operator to make the requested assignment. If the user has previously specified a vsn via a VSN control statement or if he has included the VSN keyword on the REQUEST statement, the system initiates automatic tape file assignment.

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.

The REQUEST statement can be used to create new and access existing 7- or 9-track unlabeled tapes. Although the user can also include this statement to assign a labeled tape to his job, he cannot use it to create or verify tape labels. It is suggested that LABEL statements be used for all tapes whenever possible. The default track type may be set by installation to either MT or NT.† If a track type other than the default is desired, it must be specified.

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)

†Contact installation personnel for the default track type.

- 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(lfn1, CB)
  REQUEST(lfn2, 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.

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.

VSN STATEMENT

The control statement format is:

VSN(lfn_1=vsn_1, lfn_2=vsn_2, . . . , lfn_n=vsn_n)

lfn_i Name of the file with which the specified vsn is to be associated

vsn_i 1- to 6-character volume serial number to be associated with lfn_i. If the vsn_i is zero, absent, or SCRATCH, any available scratch tape is automatically assigned to lfn_i. If characters other than letters and numbers are used, vsn_i must be specified as a literal.

The system allows tape assignment to be performed either by the system or by the operator. By supplying a vsn uniquely identifying every tape (labeled, unlabeled, and nonstandard labeled), the user enables the system to assign tapes without operator intervention.

A vsn is provided via the VSN keyword on a LABEL or REQUEST statement or via a VSN statement. 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, the user can redeclare an lfn/vsn association by returning the file. Thus, the following sequence

  ```
  VSN(FILEA=123)
  RETURN(FILEA)
  VSN(FILEA=124)
  LABEL(FILEA)
  ```

  directs the system to assign to FILEA the tape with vsn 124.

- Associate the vsns of two or more duplicate reels with one file.† If any of several duplicate reels can be used (that is, they differ only in vsns), the vsns should be separated by equal signs. Thus, the statement

  ```
  VSN(FILE1=VOL100=VOL101)
  ```

  indicates that either the tape with the vsn of VOL100 or the tape with the vsn of VOL101 can be assigned to FILE1.

- Specify the vsns of a multireel file.† If the file extends to more than one reel, the vsns for all reels required must be separated by slashes. The system assigns the reels in the order indicated in the statement. For example, the statement

  ```
  VSN(FILE2=VSN23/VSN24/VSN25)
  ```

  indicates that FILE2 may extend to the three reels identified by the vsns of VSN23, VSN24, and VSN25.

The system processes tape requests as follows:

1. Whenever a tape is mounted, the system checks for labels. If the tape was labeled, the system keeps a record of the vsn read from VOL1 and the equipment on which the tape is mounted.

2. If, when a request is made for tape assignment, an lfn/vsn association is encountered, the system compares the vsn associated with the file (or one of its equivalences) 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 required vsn is mounted. For a mounted, unlabeled tape, the operator enters a VSN command specifying the required vsn. The system is then able to automatically assign the tape.

3. If no lfn/vsn association is encountered when the request is made, the system directs the operator to assign an available unit.

4. For an ASSIGN statement, the method of assignment depends on the nn parameter. If nn is a device type (MT or NT), the operator must assign an available unit. If nn is the EST ordinal of a tape unit, the system automatically assigns the specified unit.

† Up to 55 vsns can be specified for a single file in any combination of duplicate reel and/or multireel configurations.

The following is a summary of the system and/or operator action taken in response to an ASSIGN, LABEL, or REQUEST statement. The VSN column indicates whether or not the user has specified an lfn/vsn association via the VSN keyword or a VSN statement. The mode column shows the mode as determined by the system in checking for labels.

Statement	VSN	Mode	Action
ASSIGN	Yes	Labeled	If the nn parameter is MT or NT, the operator must assign an available unit. If the nn parameter is the EST ordinal of a tape unit, assignment is automatic.
	Yes	Unlabeled	Same as when the vsn is specified and the tape is labeled.
	No	Labeled	Same as when the vsn is specified and the tape is labeled.
	No	Unlabeled	Same as when the vsn is specified and the tape is labeled.
REQUEST	Yes	Labeled	The system matches the vsn read from VOL1 with the vsn on the REQUEST or VSN statement. Assignment is automatic.
	Yes	Unlabeled	The operator enters a VSN command specifying the vsn included on the REQUEST or VSN statement.† Assignment is automatic.
	No	Labeled	The operator assigns an available unit to lfn.
	No	Unlabeled	The operator assigns an available unit to lfn.
LABEL	Yes	Labeled	The system matches the vsn read from VOL1 with the vsn on the LABEL or VSN statement. Assignment is automatic.
	Yes	Unlabeled	The operator enters a VSN command specifying the vsn included on the LABEL or VSN statement.† Assignment is automatic.
	No	Labeled	The operator assigns an available unit to lfn.
	No	Unlabeled	The operator assigns an available unit to lfn.

The LB keyword is not used in assigning a tape. Rather, it is used in processing the data on the tape once the assignment has been made.

†A VSN which contains special characters should not be specified in a request for an unlabeled tape. It is not possible to enter special characters via the VSN, xx, aaaaaa. operator command.

MAGNETIC TAPE FORMATS

The standard magnetic tapes used are 7-track, 1/2-inch tape and 9-track, 1/2-inch tape. Each type of tape can be written in binary or coded mode. Unless specified otherwise, tapes are assumed to be in binary mode. The user can select 200, 556, or 800 bits per inch (bpi) density for 7-track tapes or 800 or 1600 characters per inch (cpi) density for 9-track tapes, provided these densities are available with the hardware. Tape density can be specified by a LABEL, ASSIGN, or REQUEST control statement, the LABEL macro (refer to section 4, volume 2), or an IPRDECK installation option (refer to the NOS Installation Handbook). The system normally performs automatic processing of tape parity errors and end-of-tape conditions. However, the user can control the processing of these functions via the PO keyword on LABEL, ASSIGN, and REQUEST control statements or the up and ep fields of the FET (FET+1, bits 45 and 44).

DATA FORMATS

Data can be recorded on magnetic tape in any of eight formats.

Format	Description
I	Internal
SI	System internal†
X	External
S	Stranger
L	Long block stranger
E	Line image
B	Blocked
F	Foreign

The control statement user specifies the data format via the F keyword of a LABEL, ASSIGN, or REQUEST control statement. The LABEL macro user specifies the data format via FET+10, bits 30 through 35. The following is a description of the physical and logical characteristics of each format. Note that the user can define maximum block size, end-of-reel conditions, and noise for any format via control statement or FET parameters; the following description of these characteristics defines the suggested (and default) values.

† NOS/BE system default tape format.

I (Internal) Format

Characteristics — Description

Header

Labeled or unlabeled

Mode

Binary

Block size (PRU size)

Actual data block size can range from 0 to 512 (1000_8) CM words in exact multiples of CM words. All blocks except those containing labels include a 48-bit block terminator formatted as follows:

47	35	11	3 0
byte count	block number	0	ln

byte count — Total number of bytes in the block including the block terminator

block number — Number of blocks since the last HDR1 label†

ln — Level number

0	End-of-record
17_8	End-of-file

User-specified frame or character counts have no meaning.

Logical end-of-record

Any block with fewer than 512 (1000_8) CM words of data is considered a logical end-of-record. During a write operation, the level number field of the block terminator contains the level number obtained from FET+0, bits 14 through 17, or the WRITECW macro control word. During read operations, the system will return end-of-record status and the contents of the block terminator level number field. If the level number is 17_8, the system will also return end-of-file status. Some blocks may consist only of a block terminator.

Logical end-of-file

Any block consisting of only a block terminator with a level number of 17_8 is considered a logical end-of-file. The system ensures that an end-of-record will always precede an end-of-file by writing, if necessary, a block terminator with a level number of zero prior to the end of file.

Logical end-of-information

A tape mark followed by an EOF1 label is considered the end-of-information. This trailer sequence is generated by the system on labeled and unlabeled I and SI format tapes. The system issues a label content error if it encounters a tape mark without a valid label following it.

† Refer to appendix G.

Characteristics — Description

End-of-reel

Refer to option 3 under End-Of-Tape/End-Of-Reel Conditions.

Noise

Any block containing fewer than eight frames for 7-track tapes or six frames for 9-track tapes is considered noise, and therefore, ignored.

Special considerations

- All 9-track tapes are written in an even multiple of bytes.
- On all read operations, the system checks for fill status and compares the number of bytes read and the block number expected with the byte count and block number values in the block terminator. If the specified condition does not occur, the system handles it as if it were a parity error. This method is designed to prevent dropped or fragmented blocks; in general, it provides a much higher degree of reliability than any other format.

SI (System Internal) Format

Characteristics — Description

Header

Labeled or unlabeled

Mode

Binary or coded as indicated by FET+0, bit 1.

Block size (PRU size)

For binary mode, the block size can range from 0 to 512 (1000_8) CM words in exact multiples of CM words. Any block smaller than the maximum size except those containing labels will contain a 48-bit block terminator. This terminator has the same format as that for I format. For coded mode, the block size can range from 0 to 128 (200_8) CM words in exact multiples of CM words. Any block smaller than the maximum size except those containing labels will contain a 48-bit block terminator formatted as follows:

47	5 0
blanks	ln

ln Blank if level is 0

1 through 17_8 for all other levels

User-specified frame or character counts have no meaning.

Logical end-of-record

For binary mode, any block containing fewer than 512 (1000_8) CM words represents a logical end-of-record. For coded mode, any block containing fewer than 128 CM words represents a logical end-of-record. If a logical record consists of an exact multiple of 512 (binary) or 128 (coded) CM words, the block that denotes the logical end-of-record consists solely of a block terminator. During write operations, the level number field of the block terminator contains the level number from FET+0, bits 14 through 17, or the WRITECW macro control word. During read operations, the system will return end-of-record status and the contents of the block terminator level number field. If the level number is 17_8, the system will return end-of-file status.

Characteristics	Description
Logical end-of-file	Same as for I format.
Logical end-of-information	Same as for I format.
End-of-reel	Refer to option 3 under End-Of-Tape/End-Of-Reel Conditions.
Noise	Same as for I format.
Special considerations	● The system writes all 9-track tapes with 3n+2 mode. ● The system does not perform block checking via block terminators as is done for I format. ● For read and write operations on a coded 7-track tape, NOS is incompatible with NOS/BE. The system converts data from display code to external BCD on write operations and from external BCD to display code on read operations. NOS/BE converts data from external BCD to internal BCD on both read and write operations. ● For 7-track tapes, standard code conversion is performed. For 9-track tapes, no code conversion will be performed (it is written to tape in display code). ● For read operations, if a coded 7-track tape contains external BCD 1632 in byte 4 of a CM word, the system converts it to an end-of-line (0000 in display code). The converse is true for write operations. ● The FET device type is returned in NOS/BE format (refer to the description of the CIO OPEN macro in section 3, volume 2).

X (External) Format

Characteristics	Description
Header	Unlabeled
Mode	Binary
Block size (PRU size)	Actual data block size can range from 0 to 512 (1000_8) CM words in exact multiples of CM words.
Logical end-of-record	Any block containing fewer than 512 CM words represents a logical end-of-record. If a logical record consists of an exact multiple of 512 words, the block that denotes the logical end-of-record consists solely of a 48-bit block terminator.
Logical end-of-file	Tape mark

Characteristics	Description
Logical end-of-information	None
End-of-reel	Refer to option 1 under End-Of-Tape/End-Of-Reel Conditions.
Noise	Same as for I format.
Special considerations	• X-formatted tapes cannot be labeled. • All 9-track tapes are written in an even multiple of bytes.

S (Stranger) Format

Characteristics	Description
Header	Labeled or unlabeled
Mode	Binary or coded as indicated by FET+0, bit 1.
Block size (PRU size)	No explicit multiple of frames is required. The maximum block size may be specified in the mlrs field of the FET (FET+6, bits 0 through 17). If no block size is specified in the mlrs field, it is assumed to be 1000_8. The maximum block size is 1000_8 CM words. If the block size is longer than 1000_8, the tape is L format.
Logical end-of-record	On a CIO READ(010) or READSKP(020) request, each PRU is considered an end of record.
Logical end-of-file	Tape mark
Logical end-of-information	If the tape is unlabeled, there is no logical end-of-information. If the tape is labeled, the logical end-of-information is a tape mark followed by an EOF1 label.
End-of-reel	Refer to option 2 under End-Of-Tape/End-Of-Reel Conditions.
Noise	Any block containing fewer than 18 frames is considered noise, and therefore, ignored.
Special considerations	• Level numbers 1 through 16_8 are interpreted as level number 0. • Standard code conversion is performed for 7- or 9-track tapes in coded mode. • For CIO READ (010), WRITE (014), WRITER (024), and WRITEF (034) functions, a one-block (PRU) operation is performed with the unused bit count (FET+7, bits 24 through 29) taken from and returned to the FET. • The FET device type is returned in NOS/BE format (refer to the description of the CIO OPEN macro in section 3, volume 2).

L (Long Block Stranger) Format

The characteristics and descriptions are the same as for S format tapes except that if no block size is specified in the mlrs field (FET+7, bits 0 through 17), it is assumed to be LIMIT-FIRST-1.

E (Line Image) Format

Characteristics	Description
Header	Unlabeled
Mode	Coded
Block size (PRU size)	The block size cannot exceed 5120 frames. If the tape unit will not allow an odd number of frames to be written, the system will append a space. Unless the user specifies otherwise when he requests the tape, the system assumes the maximum block size is 136 frames.
Logical end-of-record	For a write operation, there is no logical end-of-record. For a read operation, end-of-record status is returned when a tape mark is encountered. An additional read operation returns end-of-file status.
Logical end-of-file	Tape mark
Logical end-of-information	None
End-of-reel	Refer to option 2 under End-Of-Tape/End-Of-Reel Conditions.
Noise	Same as for S-formatted tapes.
Special considerations	● E-formatted tapes cannot be labeled. ● For a write operation, a block of data will stop either at a zero byte (end-of-line) in byte 4 of a CM word or at the multiple of CM words (rounded up) based on the frame or character count. The system will then space-fill the buffer to the number of frames specified. Thus, the amount of data written will exactly equal the amount specified. ● For a read operation, if there is an odd number of characters, the system will space-fill the last six bits of the last byte and delete all trailing spaces. For control word reads, byte count and unused bit count will be set appropriately. For regular reads, EOL is guaranteed. ● For a control word write operation, no end-of-line processing is done. Data is blocked on tape using the specified frame count. Likewise for a control word read operation, no end-of-line processing is done; data is transferred to the user as it is read.

B (Blocked) Format

Characteristics	Description
Header	Unlabeled
Mode	Coded
Block size (PRU size)	The block size cannot exceed 5120 frames. If the tape unit will not allow an odd number of frames to be written, the system will append a space. Unless the user specifies otherwise when he requests a tape, the system will assume the maximum block size is 150 frames.
Logical end-of-record	For a write operation, there is no logical end-of-record. For a read operation, end-of-record status is returned when a tape mark is encountered. An additional read operation returns end-of-file status.
Logical end-of-file	Tape mark
Logical end-of-information	None
End-of-reel	Refer to option 2 under End-Of-Tape/End-Of-Reel Conditions.
Noise	Same as for S-formatted tapes.
Special considerations	● B-formatted tapes cannot be labeled. ● A write operation will stop either at a zero byte (end-of-line) in byte 4 of a CM word or at a multiple of CM words (rounded up) based on the frame or character count. ● For control word reads, byte count and unused bit count will be set appropriately. For regular reads, EOL is guaranteed. ● For a control word write operation, no end-of-line processing is done. Data is blocked on tape using the specified frame count. Likewise for a control word read operation, no end-of-line processing is done; data is transferred to the user as it is read.

F (Foreign) Format

Characteristics	Description
Label	Unlabeled
Mode	Binary or coded, as needed, for 7-track tapes and binary for 9-track tapes
Block size (PRU size)	The block size cannot exceed the CM buffer size. No explicit multiple of frames is required. The maximum block size must be specified at tape request time. The block size is used to determine whether to continue read or write operations based on the amount of data versus the space in the buffer. For example, if the maximum block size is 1000_8 CM words, the read operation will stop any time less than 1001_8 words remain. It is recommended that the user specify a buffer size equal to the largest block.
Logical end-of-record	None
Logical end-of-file	Tape mark
Logical end-of-information	None
End of reel	Refer to option 1 under End-Of-Tape/End-Of-Reel Conditions.
Noise	Any block containing fewer than 18 frames is considered noise, and therefore, ignored.
Special considerations	• For 7-track tapes, if a parity error is detected because the tape is being read in the opposite mode, the mode will be switched. • F-format operations are only done using control word reads and writes. On read operations, the control words are transferred to the user regardless of the operation being used. Labeled tapes that have been assigned as F format will have their labels treated as data on 7-track tapes. Labels will generate parity error on 9-track tapes, which process binary mode only.

END-OF-TAPE/END-OF-REEL CONDITIONS

The following is a description of the processing options for end-of-tape conditions. The user can select one of these options by default by specifying the data format or he can specify an option via the PO keyword on a LABEL, ASSIGN, or REQUEST control statement or the processing option field of the FET (FET+8, bits 36 through 47). In addition, the user processing option (FET+1, bit 45) gives the macro user control over end-of-reel conditions. For further information, refer to the CLOSER, REWIND, and UNLOAD macros described in section 3, volume 2.

Option	PO= Option	Description
1	I	If, during a write operation, the system senses the end-of-tape, it rewrites the block on which the EOT occurred as the first block on the following reel. No trailer information is written on the current reel. During a read operation, the block on which the EOT occurred is ignored and reading continues on the next reel. If a tape mark and the EOT are sensed at the same time, the EOT is ignored.
2	P	If, during a write operation, the system senses the end-of-tape, the system writes a trailer sequence, consisting of a tape mark, 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 to the user the block on which the EOT was sensed. 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.
3	S	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 tape marks for unlabeled tapes. The next block is written on the next reel. During a read operation, the EOT is noted and the system transfers to the user the block on which the EOT was sensed plus all following blocks until a trailer sequence (as described above) is recognized. Reading resumes on the next reel.

For options 1 and 2, the system is concerned only with the block on which the EOT is sensed. If tapes written using these options are transferred to another system, any data that occurs on the reel after this block should be ignored.

PRODUCT SET CONTROL STATEMENTS

Table 1-11-1 gives a list of widely used products supported by NOS and the basic control statement formats for these products. The parameters for the COMPASS control statement are described in volume 2 of this reference manual. The parameters for the other control statements in the table are given in this section. For the full array of products supported by NOS, consult the list of publications in the preface.

FORTRAN Extended 4, COBOL 4, COBOL 5, and Sort/Merge 4 use the CDC CYBER Record Manager for accessing files. NOS supports the indexed sequential, direct access, and actual key file capabilities of the Record Manager (refer to the CDC CYBER Record Manager Reference Manual).

TABLE 1-11-1. PRINCIPAL PRODUCTS SUPPORTED BY NOS

Product Name	Basic Control Statement Format
ALGOL 4	ALGOL.
BASIC 3	BASIC.
COBOL 4	COBOL.
COBOL 5	COBOL5.
COMPASS 3	COMPASS.
FORTRAN Extended 4	FTN.
Sort/Merge 4	SORTMRG.

USER LIBRARIES

NOS offers the user the option of specifying a library other than the product set default library. The user can then write library routines to perform special functions to meet his own requirements.† Routines can also be given names identical to routines from another library without causing a system conflict. This enables a user to compare the performance of library routines without modifying his software.

† Refer to the CDC CYBER Loader Reference Manual for information about the generation of a user library.

The libraries from which externals are to be satisfied can be specified as parameters on the LDSET statement as follows:

LDSET(LIB=lib_1/lib_2/.../lib_n)
LOAD(lfn)

lib_i	Library from which externals are to be satisfied. The system checks through the specified libraries sequentially.
lfn	Name of the file to be loaded.

Libraries can also be specified by using the LIBRARY statement to define the global library set.† The default system library, SYSLIB, is used to satisfy the externals if no library is specified or if unsatisfied externals exist after using the libraries specified or defaulted.

CONTROL STATEMENT FORMATS

The following is a description of the program call statements for the product set members listed in table 1-11-1 (with the exception of COMPASS, which is given in volume 2).

NOTE

Product set format does not allow file names beginning with a numeric character (refer to Control Statement Format, section 5).

† Refer to the CDC CYBER Loader Reference Manual for information about the generation of a user library.

ALGOL STATEMENT

The ALGOL control statement is used to call the ALGOL 4 compiler to a control point. The minimum memory requirement for ALGOL 4 is 46,000 octal locations. External references are satisfied from ALGOLIB.

The control statement format is:

ALGOL, p_1, p_2, . . . , p_n.

The following parameters may be supplied. The absence of any parameter suppresses the corresponding option.

p_i	Description
A	Specifies that the assembly language form of the object code is to be written on the file specified by the L option.
A=0	No assembly language listing.
A omitted	Same as A=0.
B	The output object program is to be written on file LGO.
B=lfn$_1$	The output object program is to be written on file lfn$_1$.
B=0	No binary object program.
B omitted	Same as B=LGO.
C=n	Comments interpretation for special delimiters. This option requires the compiler to search comments for special delimiters interpretation.

n	Description
0	No comments interpretation.
1	Debugging directives which are present in comments are detected by the compiler and cause debugging code to be inserted into the object program.
2	Overlay directives which are present in comments are detected by the compiler and cause overlay directives in loader input format to be inserted into the object program.
3	Array bound checking directives which are present in comments are detected by the compiler.

Multiple selection for the C option can be performed by separating each value by a slash. For example, C=3/2/1 is acceptable.

C omitted	Same as C=0.
D	The symbol file is created on file DUMPFIL.
D=lfn$_2$	The symbol file is created on file lfn$_2$.

p_i	Description
D=0	The symbol file is suppressed.
D omitted	Same as D=0.
E	The job is aborted if a fatal error occurs during compilation. If an EXIT control statement is included in the job, exit processing is performed.
E=0	Suppresses abort in the event of a fatal error.
E omitted	Same as E=0.
F	If a fatal error is found in the first pass, compilation is terminated at the end of this pass.
F=0	Continue until the normal end of compilation.
F omitted	Same as F=0.
G	Compilation will consider stack swapping to ECS and when the program is executed, the swapping procedures are activated automatically.
G=0	No swapping will be considered.
G omitted	Same as G=0. This option must not be selected when using a machine without ECS; otherwise, unpredictable results leading to a fatal error are obtained.
I	Source input is on file INPUT.
I=lfn_3	Source input is on file lfn_3.
I=0	No source input.
I omitted	Same as I=INPUT.
K=n	Input record size. n The number of significant characters to be interpreted by the compiler on the source statement image.
K omitted	Same as K=72.
L	The source program is listed with fatal diagnostics on file OUTPUT.
L=lfn_4	The source program is listed with fatal diagnostics on file lfn_4.
L=0	Fatal diagnostics only are listed on file OUTPUT.
L omitted	Same as L=OUTPUT.
N	A listing of advisory diagnostics is generated on the file specified by the L option.
N=0	Advisory diagnostics are suppressed; only diagnostics fatal to code generation are listed.

p_i	Description
N omitted	Same as N.
O=n	Specifies the level of compiled optimization.

n	Description
0	The program is compiled in fast compile mode.
1	Linguistic optimization is performed by optimizing procedure calling.
2	Optimizations of O=1 and also subscript and for statement optimizations.

O	Same as O=0.
O omitted	Same as O=0.
P	Specifies that the assembly language form of the object code is to be punched in standard assembly language card format on file PUNCH.
P=lfn$_5$	Assembly language is to be punched on file lfn$_5$.
P=0	Assembly language punching is suppressed.
P omitted	Same as P=0.
Q	Creates interactive file on file QFILE for ALGOL Interactive Debugging Aids (AIDA).
Q=lfn$_6$	Creates interactive file on file lfn$_6$.
Q=0	Suppresses interactive compilation and file.
Q omitted	Same as Q=0. Q may not be specified if the S option is selected.
R	A cross-reference map is produced and listed at compile time for identifiers in the source program on the file specified by the L option.
R=0	No cross-reference map is produced.
R omitted	Same as R=0.
S=n	Array storage location.

n	Description
0	All arrays are allocated to CM.
1	Virtual arrays are allocated to ECS.
2	All arrays are allocated to LCM. This option applies only to programs executed on a CDC CYBER 70 Model 76.

S may not be specified if the Q parameter is specified.

p_i	Description
S omitted	Same as S=0.
U	Specifies that the file COMPILE contains user implicit outer block head input, supplementary to the file specified with the I option.
U=lfn$_7$	The source program is preceded by the implicit outer block head list on file lfn$_7$.
U=0	There is no file for implicit outer blocks.
U omitted	Same as U=0.
X	Allows real-integer (or integer-real) correspondence between formal and actual parameters; if real-integer, perform the conversion.
X=0	Forbids any real-integer (or integer-real) correspondence between formal and actual parameters. Selection of this option significantly degrades performance of the program.
X omitted	Same as X.

BASIC STATEMENT

The BASIC control statement is used to call the BASIC 3 compiler to a control point. The minimum memory requirement for BASIC 3 is 35000 octal locations. Ordinarily, a BASIC program is compiled in place. Since all object-time routines are contained within the compiler, no external references are generated. However, the user can include the B parameter to generate relocatable code. When this code is loaded, externals are satisfied from the library BASLIB.

The control statement format is:

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

The following parameters may be supplied.

p_i	Description
AS	Source program is encoded in extended ASCII character set (program will run in ASCII mode).
AS=0	Source program and data files contain only normal (non-ASCII) characters.
AS omitted	Same as AS=0.
B	Relocatable object code is written on file BIN (this option requires at least 4000_8 words of additional memory).
B=lfn	Relocatable object code is written on file lfn (this option requires at least 4000_8 words of additional memory).
B=0	Specifies compilation to memory; no relocatable object code is generated.

p_i Description

B omitted Same as B=0.

BL Page eject between source, object, and execution output listing. This option is ignored if output is returned to a time-sharing terminal.

BL omitted Page eject between source and object listing is suppressed; listing is not burstable. Unless the B parameter is omitted, page eject preceding execution output is also suppressed.

E Compiler error diagnostics are written on file ERRS.

E=lfn Compiler error diagnostics are written on file lfn.

E omitted Compiler error diagnostics are written on the file specified by the L parameter. If L=0, they are written on the file OUTPUT.

EL=W Warning diagnostics and fatal compiler diagnostics are written on the file specified by the E parameter.

EL=F Fatal compiler diagnostics are written on the file specified by E parameter; no warning diagnostics are included.

EL omitted Same as EL=W.

GO Executes compiled BASIC program (if B is specified, the relocatable binary is loaded and executed; if B is omitted, the compiled-to-memory code is executed).

GO=0 Inhibits execution; neither compiled-to-memory nor relocatable binary code will execute.

GO omitted Compiled-to-memory code is executed. Relocatable binary code (B specified) is not executed.

I Source input is on file COMPILE. If I is omitted, input is on file INPUT.

I=lfn Source input is on file lfn.

I omitted Input will come from the file INPUT.

J=lfn Run-time input is on file lfn.

J=0 No run-time input file is used. If this option is specified, use of the INPUT statement aborts an executing BASIC program.

J omitted Run-time input is on file INPUT.

K Execution output is written on file OUTPUT.

K=lfn Execution output is written on file lfn.

K omitted Same as K.

L Listable compiler output is written on file OUTPUT.

p_i	Description
L=lfn	Listable compiler output is written on file lfn.
L=0	No listable compiler output is generated.
L omitted	For batch origin jobs, this is the same as L. For time-sharing origin jobs, listable compiler output is suppressed.
LO	Source listing is written on file specified by the L parameter.
LO=S	Source listing is written on file specified by the L parameter.
LO=O	Object code and source listing is written on file specified by the L parameter.
LO=0/O	Object code listing is written on file specified by the L parameter.
LO=0	Turns off all list options.
LO omitted	Same as LO.
PD or PD=8	Sets print density to 8 lines per inch for files specified by the K and L parameters. The installation default print density is automatically reset after output is written.
PD=6	Sets print density to 6 lines per inch for files specified by the K and L parameters. The installation default print density is automatically reset after output is written.
PD omitted	Sets print density for files specified by the K and L parameters to the installation default (usually 6).
PS=n	Specifies page size as n printable lines per page ($4 \leq n \leq 32768$).
PS omitted	If PD is also omitted or specifies a print density default, the installation default page size will be used. If PD specified a nondefault print density, PS is calculated with the formula: PS=PD *(default-PS)/(default PD)

COBOL STATEMENT

The COBOL control statement calls the COBOL 4 compiler to a control point. The minimum memory requirement for COBOL is 52000 octal locations. External references are satisfied from COBOL.

The control statement format is:

COBOL, p_1, p_2, . . . , p_n.

If the control statement does not fit on one card, it can be continued on the next card. However, the last character on the first card must be a separator, such as (,+- or /.

The following parameters can be supplied.

p_i	Description
A	Leading blanks are treated as zeros in arithmetic statements and comparisons.
B	Relocatable binary object code is written to file LGO. If the B parameter is omitted, this option is assumed.
B=lfn$_1$	Binary object code is written to file lfn$_1$. The file name should specify a mass storage file.
B=0	Suppresses binary output of object code.
B omitted	Relocatable binary file is writen on file LGO.
BUF	In COBOL 4, buffer sizes are based on the record description; a minimum size of 514 words has been established. The BUF parameter selects the version 3 method which does not use record description or ALTERNATE AREAS to determine the minimum block size.
C	Specifies that a copy is to be made from source rather than from the library, which is the default condition.
D	Inhibits COBOL program execution when an E diagnostic is encountered.
DB	Checks for subscript range errors. If an error is encountered, the run is terminated. If DB is not specified, no check is made for subscript range errors.
DB1	Allows generation of object code which calls paragraph trace feature to trace the flow of the COBOL program.
E	Allows output of a COBOL compilation to be added to the system library with EDITLIB. This parameter has no application for NOS.
E=program-name	The main overlay of the program is named by program-name, which must not exceed five characters.
F	All data name entries described as COMPUTATIONAL are interpreted as COMPUTATIONAL-1 when this parameter is included.
H	Increases sort efficiency if no OPEN statement is executed during SORT input or output procedure. If this parameter is not used, unnecessary space is reserved for all program files. If a file is opened during input/output and H is specified, the run is terminated.
I	Specifies that compiler input is to be obtained from file INPUT.
I=lfn$_2$	Compiler input is obtained from file lfn$_2$. Tape files must be BCD.
I omitted	Same as I=INPUT.
L	Specifies that the listing is to be written on file OUTPUT.
L=lfn$_3$	Output is to be written on file lfn$_3$.

p_i Description

L=0 Suppresses output except for errors.

The L parameter may appear with one of the following suffixes to produce special listings.

Suffix	Meaning
C	Listing of items copied from user libraries
M	Data map
O	Object code in octal
R	Data-name and procedure-name cross-reference list with pointers to source lines
X	Extended diagnostics

L omitted Same as L=OUTPUT.

N Directs the COBOL 4 compiler to issue an E type diagnostic if a non-ANSI feature is detected.

OB Separates binary overlay segments† from main program and writes them on LGO.

OB=lfn_4 Specifies that binary output from overlay segments† is to be written on lfn_4.

OB omitted Same as OB.

P Allows the user to execute a strict ANSI program.

S Specifies that external references are to be satisfied from the source library file COLIB.

S=filenam External references are to be satisfied from filenam, which contains the COBOL source library.

S omitted Same as S.

SUB Suppresses data division binary output that duplicates output from a separately compiled main program, except for working storage and constant sections, so that the subprogram and main program can be loaded together.

SUBM Identifies the COBOL program as a subprogram so that it can be called from a main program written in another language.

T Requests a tape sort rather than a disk sort. This requires four files which may be assigned to the disk.

U Allows use of a collating sequence other than the standard Control Data sequence.

†NOS does not support segmentation.

p_i Description

V If the loaded program is to be saved using NOGO with the file name specified, the V parameter must be specified for all COBOL/SORT programs. Specifying this parameter causes the SORT code to be included in the program rather than being loaded dynamically.

W An independent segment† (priority number 50 through 99) may overlay or be overlaid by an overlayable fixed segment or another independent segment. In COBOL version 4 and for ANSI programs, an independent segment is made available in its initial state. To override this usage and provide independent segments in their last used state so that COBOL 3 programs can be run without change, the W parameter must be included.

Z Ensures compatibility with COBOL version 3 and turns on the C and W parameters.

COBOL5 STATEMENT

The COBOL5 control statement calls the COBOL 5 compiler to a control point. External references are satisfied from the library COBOL 5.

The control statement format is:

COBOL5, $p_1, p_2, \ldots, p_n$.

This control statement cannot be continued. The following parameters are supplied.

p_i Description

ANSI Equivalent to ANSI=T.

ANSI=s Language extensions that do not conform to ANSI X3.23-1974, COBOL are diagnosed and treated as errors with severity specified by s.

T Trivial error

F Fatal error

The EL=T parameter must be specified to obtain a listing of diagnostics that note language extensions.

ANSI omitted Language extensions that do not conform to ANSI X3.23-1974, COBOL are allowed.

APO The ASCII apostrophe character with a display code value of 70 (Hollerith 11-8-5 punch, sometimes punched by an up arrow key) delimits nonnumeric literals in the source program instead of the quotation mark character of display code value 64 (Hollerith 8-4 punch, sometimes punched by a not equal sign).

APO omitted Nonnumeric literals in the source program are delimited by the quotation mark character that has a display code value of 64.

† NOS does not support segmentation.

Pi	Description
	This option reverses the action of the ' and " so that the ' can be used within an alphanumeric literal the same as any other character. Within a source program, this option can be selected by the QUOTE IS APOSTROPHE clause of the SPECIAL-NAMES paragraph.
B	Binary output from compilation is written to file BIN.
B=0	No binary output is produced during compilation.
B=lfn	Binary output from compilation is written to file lfn, with lfn being one through seven letters or digits beginning with a letter.
B omitted	Binary output from compilation is written to file LGO.
BL	Page eject occurs between various parts of the listing.
BL omitted	Triple space separates the program listing, diagnostics, cross-reference listing, and any cross-reference map.
CC1	Data items described as COMPUTATIONAL are stored and processed as COMPUTATIONAL-1 items. Selection of this parameter allows programs written for other compilers to gain the efficiencies of COMP-1 processing.
CC1 omitted	Data items described as COMPUTATIONAL are stored and processed as COMPUTATIONAL items.
CPY	COPY statements in the source program are compiled using the text on the UPDATE random program library file declared with the X parameter.
CPY omitted	COPY statements do not appear in the source program; if they appear, they generate a fatal compilation error.
D	Equivalent to D=SSFILE.
D=lfn	Subschema for the CDCS interfaces resides on file lfn, where lfn is one through seven letters or digits beginning with a letter.
D omitted	Subschema for the CDC CYBER Database Control System is not used; if the SUB-SCHEMA appears in the program, it generates a fatal compilation error.
DB	Equivalent to DB=DL/SB/B.
DB=B	Binary executable code is produced regardless of all errors in the source program. Lines with errors of severity C or F result in compilation of a call to an execution time abort routine; execution of those lines aborts the program. If DB=B is not selected, the first occurrence of a C or F error inhibits generation of executable code.
DB=DL	Debugging lines in the source program (lines with a D in column 7) are compiled as executable code.

P_i	Description
	If DB=DL is not selected, all debugging lines are treated as comment lines, unless the WITH DEBUGGING MODE clause appears in the program. The presence of the WITH DEBUGGING MODE clause causes the DL option to be ignored.
DB=SB	Code compiled such that subscript and index references are checked during execution to ensure that all references to tables are within the table bounds. An out-of-bounds reference aborts the program with a dayfile message that identifies the line with the incorrect reference. If DB=SB is not selected, subscripted and indexed references are not checked during execution.
DB=TR	Paragraph trace occurs during execution. If DB=TR is not selected, paragraph trace does not occur, and any references to the trace directives called by ENTER statements result in a fatal error.

Multiple options for the DB parameter can be specified by separating the options with a slash.

DB omitted or DB=0	None of the debugging options applicable to this parameter are selected.
E	Error information specified by the EL parameter is written to the file ERRS.
E=lfn	Error information specified by the EL parameter is written to the file with the name lfn, where lfn is one through seven letters or digits beginning with a letter.
E omitted or EL=0	Error information specified by the EL parameter is written to the file OUTPUT.

When the L parameter specifies full listing, information written to the file specified by the E parameter is also written to the file specified by the L parameter. If the lfn specified by the E parameter is the same as that specified by the L parameter, information is written only to the error file.

EL	Equivalent to EL=W.
EL=T	Trivial errors plus all errors of levels W, F, and C are listed. Level T errors indicate a suspicious usage; although the syntax is correct, the usage is questionable. EL=T is required to obtain a listing of the messages reported as UNLISTED NON-ANSI ERRORS on the diagnostic summary.
EL=W	Warning errors plus all errors of levels F and C are listed. Level W errors indicate the syntax of the statement is incorrect and the compiler has made an assumption and continued compilation.
EL=F	Fatal erros plus all level C errors are listed.

p_i Description

Level F errors indicate an error that prevents compilation of the statement. Unresolvable semantic errors and propagated errors caused by earlier level F errors are among the causes of level F errors.

EL=C Catastrophic errors are listed.

Level C errors are fatal to compilation of the current program. Compilation resumes at the Identification Division header of any program immediately following without an intervening file boundary.

EL omitted Levels F and C errors are listed.

Errors are listed on the file specified by the E parameter.

ET=opt The compiler aborts if the executable code contains any errors of at least the T, W, F, or C severity indicated by opt. Levels are those indicated by the EL parameter.

Level T or W errors produce executable binary code. Level E and C errors produce a short, bad, binary program that causes the loader to inhibit loading, unless the B option of the DB parameter is specified.

The job resumes after any EXIT(S) control statement in the job stream.

ET omitted The next control statement in the job stream is executed after termination, despite any errors diagnosed during compilation.

I Card images of program to be compiled reside on file COMPILE.

I=lfn Card images of program to be compiled reside on file lfn, where lfn is one through seven letters or digits beginning with a letter.

I omitted Card images of program to be compiled reside on file INPUT.

L Source listing, diagnostics, and information selected by the LO parameter are written to file LIST.

L=0 No listing is produced.

L=lfn Source listing, diagnostics, and information selected by the LO parameter are written to file lfn, where lfn is one through seven letters.

L omitted Source listing, diagnostics, and information selected by the LO parameter are written to file OUTPUT.

LBZ All leading blanks in numeric fields are treated as zeros in arithmetic statements and comparisons.

Selection of this parameter significantly slows execution time and increases the size of compiled code.

LBZ omitted Numeric fields that contain blanks are in error.

LO Equivalent to LO=S/M/R.

p_i Description

LO=0 None of the information that can be selected by O, R, M, or S is listed.

LO=M A map that correlates program entities, attributes such as data class and size, and physical storage is listed.

LO=O Generated object code with COMPASS mnemonics is listed.

LO=R Cross-reference of program entities and locations of definitions and use within the program are listed.

LO=S Source program is listed.

Multiple options for the LO parameter can be selected by separating the options with a slash.

LO omitted Equivalent to LO=S.

MSB Program is compiled as a subroutine that includes COBOL initiation.

This parameter should be used only when the COBOL program is called by a program written in a language other than COBOL. It should not be used for a COBOL subprogram that is called by another COBOL program. Only the first COBOL program called in a group of independently compiled subprograms should specify MSB.

MSB omitted Normal program is compiled.

PD Equivalent to PD=8.

PD=3 Listing specified by L and E parameters is double spaced at six lines per inch.

PD=4 Listing specified by L and E parameters is double spaced at eight lines per inch.

PD=6 Listing specified by L and E parameters is single spaced at six lines per inch.

PD=8 Listing specified by L and E parameters is single spaced at eight lines per inch.

PD omitted Equivalent to PD=6.

The PD parameter is ignored for connected interactive terminal listings. Any option specified by this parameter must be supported by the printer on which the files are output.

PS=n Number of lines on a printed output page is n.

Three lines exist at the top and at the bottom of each page, in addition to n.

PS omitted Number of lines on a printed output page is the density specified by [(PD parameter) multiplied by (IP.PS/IP.PD)], where IP.PS and IP.PD are two installation parameters.

pi — Description

PW — Lines of printed output are 72 characters in length.

PW=n — Lines of printed output are n characters in length. The compiler reformats listing lines to this length.

PW omitted — Lines of printed output are 136 characters in length.

SB — Program is compiled as a subprogram. If the main program is not written in COBOL, the MSB parameter must also be used for one of the subprograms.

SB omitted — Program is compiled as a main program.

SY — Source program is checked for syntax, but executable code is not generated.

When SY is selected, compilation time is approximately half that required when SY is omitted.

SY omitted — Source program is compiled and executable code is generated.

U — Equivalent to U=COMPS.

U=lfn — COMPASS line images of the generated program are written to file lfn in a format acceptable for the UPDATE utility, where lfn is one through seven letters or digits beginning with a letter.

The first seven characters from the name in the PROGRAM-ID paragraph become the deck name on a *DECK image written as the first item on the file. The second image on the file is *IDENT with the same name as the deck name.

U omitted or U=0 — COMPASS assembly language images are not produced.

UC1 — All COMP-1 items are converted to integer format before they are processed.

Conversion occurs through the use of an unpack instruction that removes the exponent. UC1 should be used only when files created by COBOL 4 are being processed under COBOL 5. COMP-1 items in COBOL 4 have a different format in COBOL 5. Larger and slower object programs result from this parameter.

UC1 omitted — All COMP-1 items are processed in COMP-1 format.

X — Equivalent to X=NEWPL.

X=lfn — The UPDATE random program library containing text for COPY statements is on file lfn, where lfn is one through seven letters or digits beginning with a letter.

X omitted or X=0 — Equivalent to X=OLDPL.

FTN STATEMENT

The FTN control statement calls the FORTRAN Extended compiler to a control point. The minimum memory requirement for FORTRAN Extended is 46,000 octal locations. FTN externals are satisfied from the user library FORTRAN.

The control statement format is:

FTN, p_1, p_2, . . . , p_n.

p_i	Description
A	Branches to EXIT control statement if fatal compilation error occurs. If there is no EXIT statement, the job terminates.
A=0	Control transfers to the next control statement, regardless of the installation default, if fatal compilation errors occur.
A omitted	Same as A=0.
B	Object code is written in file LGO.
B=lfn$_1$	Object code is written on file lfn$_1$.
B=0	Suppresses object code output.
B omitted	Same as B=LGO.
BL	Generates output listing that is easily separable into components by issuing page ejects between source code, error summary (if present), cross-reference map, and object code (if requested), and ensures that each program unit listing contains an even number of pages (page parity) by issuing a blank page at the end if necessary.
BL=0	Listings are produced in compact format.
BL omitted	Same as BL=0.
C	Uses COMPASS assembler for compiler-generated code.
C=0	Selects the FTN assembler regardless of the installation default.
C omitted	Same as C=0.

The C option conflicts with the E, Q, and TS options.

D	Debug mode of compilation; a minimum of 61,000$_8$ locations is required if this option is selected. Debug input is obtained from INPUT source.
D=lfn$_2$	Debug input is obtained from lfn$_2$.
D=0	Debug statements are ignored.
D omitted	Same as D=0.

The D option conflicts with the TS option.

p_i — Description

E — Compiler-generated object code on file COMPS is output as COMPASS line images for input to Update.

E=lfn$_3$ — Compiler-generated object code on lfn$_3$ is output as COMPASS line images for input to Update.

E=0 — Normal binary object file is generated.

E omitted — Same as E=0.

EL=ℓ — Lists diagnostics according to list specification ℓ:

ℓ	Description
A	Lists diagnostics indicating all non-ANSI usages, as well as fatal diagnostics. Also, lists informative diagnostics if compiling under OPT=0, 1, or 2; lists note and warning diagnostics if compiling in TS mode.
I	Lists informative and fatal diagnostics if compiling under OPT=0, 1, or 2; lists note, warning, and fatal diagnostics if compiling in TS mode.
N	Lists note, warning, and fatal diagnostics if compiling in TS mode; lists fatal diagnostics if compiling under OPT= 0, 1, or 2.
W	Lists warning and fatal diagnostics if compiling in TS modes; lists fatal diagnostics if compiling under OPT= 0, 1, or 2.
F	Lists fatal diagnostics.

EL omitted — Same as EL=I.

ER — Includes code for object time reprieve.

ER=0 — No object time reprieve code is included.

ER omitted — The same as ER if TS=0 or OPT=0. The same as ER=0 if OPT=1 or 2.

G — Loads the first system text overlay from the sequential binary file SYSTEXT. A maximum of seven system texts can be specified by any combination of the G, S, and C parameters.

G=lfn$_4$ — Loads the first system text overlay from the sequential binary file lfn$_4$.

G=lfn$_4$/ovl — Searches the sequential binary file, lfn$_4$, for a system text overlay with the name ovl and loads the first such overlay encountered.

G=0 — No system text is loaded.

G omitted — Same as G=0.

GO — Binary object file is loaded and executed at end of compilation.

GO=0 — Binary object file is not loaded and executed.

p_i Description

GO omitted Same as GO=0.

I Source input is on file COMPILE.

I=lfn$_5$ Source input is on file lfn$_5$.

I omitted Same as I= INPUT.

L Listable output (specified by list control options BL, EL, OL, R, and SL) is to be written onto file OUTPUT. If list control options are not specified, the listing consists of the source program, informative and fatal diagnostics, and a short reference map.

L=lfn$_6$ Listable output is to be written onto file lfn$_6$.

L=0 Fatal diagnostics and the statements that caused them are listed on the file OUTPUT. All other compile-time output, including intermixed COMPASS, is suppressed. List control options are ignored.

L omitted Same as L=OUTPUT.

LCM=D Selects 17-bit address mode for level 3 data. This method produces more efficient code for accessing data assigned to level 3. User ECS field length must not exceed 131,071 words. If the LCM parameter is omitted, this option is assumed.

LCM=I Selects 21-bit address mode for level 3 data. This mode depends heavily upon indirect addressing. LCM=I must be specified if the execution ECS field length exceeds 131,071 words. In TS mode, all LCM addressing is done in 21-bit mode, regardless of the LCM parameter.

LCM Same as LCM=D.

LCM omitted Same as LCM=D.

In time-sharing mode, all addressing is done in 21-bit mode, regardless of the LCM specification.

ML Current data in the form yyddd is used for the MODLEVEL micro.

ML=nnn Specifies nnn as the value of the MODLEVEL micro used by COMPASS. nnn consists of one to seven alphanumeric characters.

ML omitted Same as ML.

OL Generated object code is listed on the file specified by the L parameter.

OL=0 Object code is not listed.

OL omitted Same as OL=0.

p_i	Description

OPT=n	Level of optimization:
	n=0 Fast compilation (automatically activates T option).
	n=1 Standard compilation and execution (default value).
	n=2 Fast execution. OPT=2 is equivalent to OPT.

OPT omitted	Same as OPT=1.
P	Page numbering is continuous from subprogram to subprogram, including intermixed COMPASS. If P is omitted, page numbers begin at 1 for each subprogram.
P=0	Page numbers begin at 1 for each subprogram.
P omitted	Same as P=0.
PD	Same as PD=8.
PD=6	Print density 6 lines per inch.
PD=8	Print density 8 lines per inch.
PD omitted	Same as PD=6.
PL=n	n is the maximum number of records produced by the user program at execution time which can be written on the file OUTPUT. n does not include the number of records in the source program listing and compilation and execution time listings: n ≤ 999 999 999.
PL=nB	An octal number must be suffixed with a B; n ≤ 77 777 777.
PL omitted	Same as PL=5000.
PS=n	n is the maximum number of lines per page.
PS omitted	Same as PS=60.
PW	Specifies 72-character page width. This option is valid only for time-sharing origin jobs.
PW=n	Specifies page width of n characters (50≤ n≤ 136). This option is valid only for time-sharing origin jobs.
PW omitted	Same as PW=126 if the output goes to a printer. Same as PW=72 if the output goes to a terminal.
Q	Compiler performs full syntactic scan of the program, but no object code is produced. No code addresses are provided if a reference map is requested. This mode is substantially faster than a normal compilation, but it should not be selected if the program is to be executed.
Q=0	Normal compilation.

p_i Description

Q omitted Same as Q=0.

The Q option conflicts with the B, C, GO, OL, TS, and E options.

R=n Selects the kind of reference map required:

n=0 No map

n=1 Short map (symbols, addresses, properties, and a DO-loop map)

n=2 Long map (short map plus references by line number)

n=3 Long map with printout of common block members and equivalence groups

In time-sharing mode, R=3 and R-2 are the same. Common and equivalence groups are not listed.

R omitted Same as R=1.

ROUND=s Directs the compiler to produce code that rounds arithmetic operations involving the following operators: (s=*/+ or -).

ROUND=0 Computation for the indicated operators is not rounded.

ROUND Implies ROUND=+-*/

ROUND omitted Same as ROUND=0.

S=ovl The system text overlay, ovl, is loaded from the job's current library set.

S=lib/ovl The system text overlay, ovl, is loaded from lib. lib can be a user library file or a system library.

S=0 When COMPASS is called to assemble any intermixed COMPASS programs, it will not read in a system text file.

S omitted Same as S=SYSTEXT if G=0. Same as S=0 if G≠0.

SEQ Source input file is in sequenced line format. Specifying this option automatically activates the TS option.

SEQ=0 Source input file is in standard FORTRAN format.

SEQ omitted Same as SEQ=0.

The SEQ option conflicts with the OPT=0, 1, or 2 option.

SL Source program is listed on the file specified by the L parameter.

SL=0 Source program is not listed.

SL omitted Same as SL.

p_i Description

SYSEDIT All input/output references are accomplished indirectly through a table search at object time. File names are not entry points in main program, and subprograms do not produce external references to the same file.

SYSEDIT=0 Input/output references accomplished directly; file names are used as entry points in the main program, and subprograms produce external references to the file name.

SYSEDIT omitted Same as SYSEDIT=0.

This feature is used primarily for system-resident programs.

T If this parameter is specified, full error traceback occurs. This is primarily used for programs in debug stages. Selecting the D parameter or OPT=0 automatically activates the T option.

T=0 No traceback occurs when an error is detected.

T omitted Same as T=0.

TS Time-sharing mode. Compilation speed and field length are optimized at the expense of execution speed and field length. TS mode is preferable to the optimizing compilation modes (OPT-1, 2, or 3) for the debugging stages of a program. Specifying the TS option together with the C, D, E, or Q option constitutes a fatal control statement error.

TS omitted Same as specifying OPT=1.

UO This allows the compiler to perform potentially unsafe optimizations. The UO parameter is ignored unless OPT=2 is also specified.

UO=0 Unsafe optimization is not performed.

UO omitted Same as UO=0.

X File OPL is the source of external text (XTEXT) when location field of XTEXT pseudo instruction is blank. Only one X parameter may be specified.

X=lfn_7 External text on file lfn_7.

X omitted External text is on OLDPL.

This feature is for COMPASS subprograms only.

Z When Z is specified, all subroutine calls having no parameters are forced to pass a parameter list consisting of a zero word. This feature is useful to COMPASS-coded subroutines expecting a variable number of parameters. Z should not be specified unless necessary, since programs require less memory if Z is omitted.

Z=0 A zero word parameter list is not passed.

p_i Description

Z omitted Same as Z=0.

SORTMRG STATEMENT

The SORTMRG control statement calls Sort/Merge to process a logical record of directives. The minimum memory requirement for Sort/Merge is 25,000 octal locations.

The control statement format is:

SORTMRG(p_1, p_2, . . . , p_n)

p_i Description

I Sort/Merge directives are on file COMPILE.

I=lfn$_1$/r$_1$ Sort/Merge directives are on file lfn$_1$ with the following rewind options.

r_1 Description

R File is rewound before opening. If system INPUT file is indicated, R should not be specified.

NR File is not rewound before opening. This option is assumed if r_1 is not specified.

I omitted Same as I= INPUT.

MO=n Intermediate merge order, $2 \leq n \leq 64$. In general, higher merge orders produce faster sorts at the expense of greater field length requirements. If insufficient core is available to merge at the requested order, a fatal error occurs, and a diagnostic indicates how much additional core is required.

MO=*n Intermediate merge order, $2 \leq n \leq 64$. If insufficient core is available to merge at the requested order, merge will take place at a smaller order, and an informative diagnostic is issued.

MO omitted The installation default merge order is used (release system default is 5).

O Listings will be written to the file OUTPUT.

O=lfn$_2$/r$_2$ Listings are written to file lfn$_2$ with the following rewind options.

r_2 Description

R File is rewound before opening. If system OUTPUT file is indicated, R should not be specified.

NR File is not rewound before opening. This option is assumed if r_2 is not specified.

O omitted Same as O=OUTPUT.

OWN Owncode binaries are on file LGO. If the OWN parameter is omitted, owncode binaries are on file INPUT.

<table>
<tr><th>p_i</th><th>Description</th></tr>
<tr><td>OWN=lfn$_3$/r$_3$</td><td>Owncode binaries are located on file lfn$_3$ with the following rewind options.
<table>
<tr><th>r_3</th><th>Description</th></tr>
<tr><td>R</td><td>File is rewound before opening. If system INPUT file is indicated, R should not be specified.</td></tr>
<tr><td>NR</td><td>File is not rewound before opening. This option is assumed if r_3 is not specified.</td></tr>
</table></td></tr>
<tr><td>OWN omitted</td><td>Owncode binaries are on file INPUT.</td></tr>
</table>

CHECKPOINT/RESTART 12

A job may be terminated at any time 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, section 3.

CKP STATEMENT

The CKP control statement causes a checkpoint dump to be taken.

The control statement format is:

CKP(lfn$_1$, lfn$_2$, . . . , 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 will be 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 statements with the CK 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.
- 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 will be 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 will be 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 will be unable to retrieve that file and will abort. 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=4095, the numbering sequence begins again at nnnn=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 CB was specified when the checkpoints occurred, each dump is written over the preceding dump, and therefore, only the last dump is available. If two REQUEST, ASSIGN, or LABEL statements are submitted, successive CB-type dumps are alternated between two files; therefore, the last two dumps are available.†

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.

†If alternate checkpoint files are used and a read parity error occurs in an attempt to read the last checkpoint, RESTART will abort even if the NA option was selected.

This section contains a description of central memory dumps and their use as a debugging aid. This information should be of considerable assistance to the user in finding errors in his program.

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 1-13-10 is an example of a central memory dump. Section 9 describes the options of the DMP control statement that can be used to obtain various dumps.

The user may also dump his exchange package. Figure 1-13-1 illustrates the format of the actual exchange package.

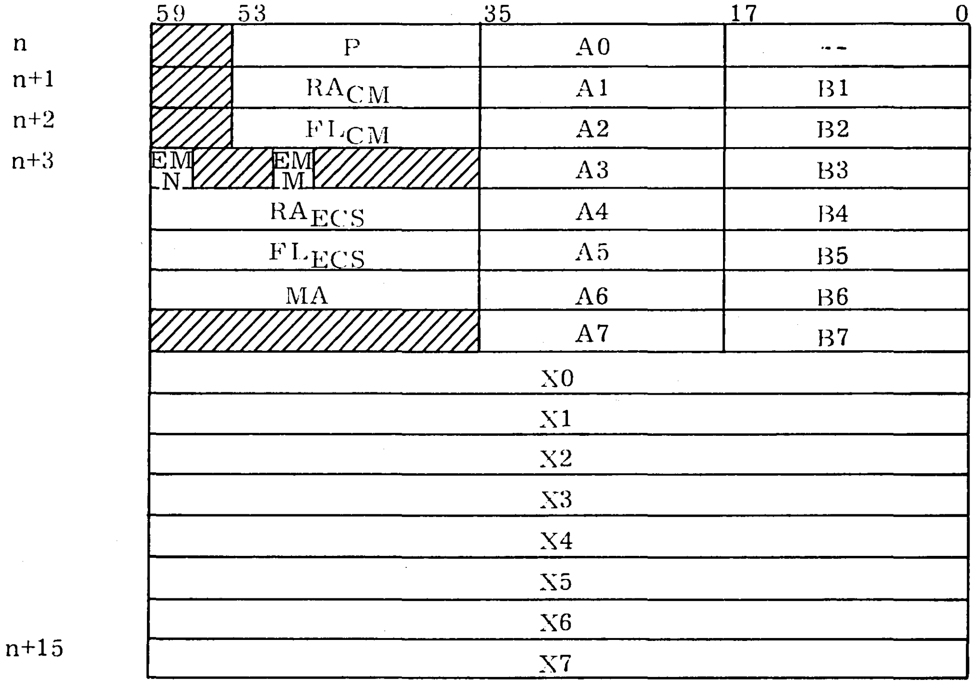


Figure 1-13-1. Exchange Package

P	Program address currently being executed
RA	Reference address, beginning address of the associated field length [central memory (CM), extended core storage (ECS)]
FL	Field length
EM-M	Program error exit mode (refer to section 6)
EM-N	Hardware error exit mode (refer to section 6)
MA	Monitor address
An	Address registers
Bn	Increment registers
Xn	Operand registers

When the user requests this form of a dump, he also receives the following information.

- The contents of memory at the address contained by the A registers, identified as (An)
- The contents of RA (reference address) and RA+1, identified as (RA) and (RA+1), respectively
- 40_8 locations before and after the address contained in P (100_8 locations total)

Figure 1-13-9 illustrates an example of this exchange package dump.

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 statements 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 1-13-2 through 1-13-10 are output from a FORTRAN program source deck processed by the following sequence of control statements.

```
TEST(CM50000,T10)
USER(ABCD,PASS,FAMA)
SETCORE,0.
MAP.
FTN.
LGO.
OVLA.
DMP.
DMP,1000.
```

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, common blocks, etc. (refer to figure 1-13-2).

Figures 1-13-6 and 1-13-7 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 1-13-8).

Figures 1-13-9 and 1-13-10 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 1-13-2, the variable I is used as the control variable in the DO loop defined by statements 10 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 1-13-2), noting I is at relative address 4167_8.
2. Find the first word address (FWA) of the main overlay TESTA. (Refer to the load map, figure 1-13-6.) The FWA of TESTA is 143_8.
3. Add ($143_8 + 4167_8 = 4332_8$) to obtain the absolute address of I.
4. In figure 1-13-10, address 4332 contains 0013_8 (11_{10}). This should be 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 1-13-2). The value is 12, which means that B begins at relative location 12_8 of common block AAA. By referring to the map (figure 1-13-6), note that AAA begins at absolute address 101_8. Therefore, $101_8 + 12_8$ (relative location of B) equals 113_8, the beginning address of array B. B(1) is 113_8, and the address of B(3) is 115_8.
- The location in core of the B array is illustrated in figure 1-13-10.

Example 3: (Finding an Address Within the Program)

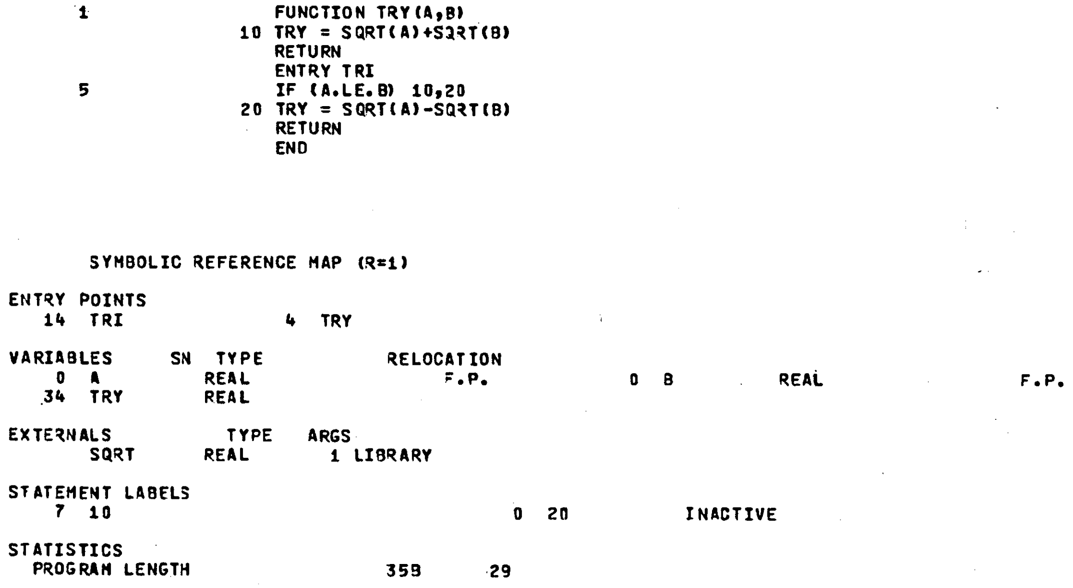
Referring to figure 1-13-9, note that the program stopped at address 10114 (the value of P). To find where this is in the program, the following points must be considered.

- Figure 1-13-6 or 1-13-7 contains the routine addresses.
- Figure 1-13-6 illustrates that routine SYS.RM is at address 10114. This means the program ended in routine SYS.RM.

```
      OVERLAY(OVLA,0,0)
       PROGRAM TESTA(INPUT,OUTPUT)
      COMMON//E(100)
      COMMON/AAA/A(10),B(10),C(3,3)
      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./
      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)
      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))
      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
      END
```

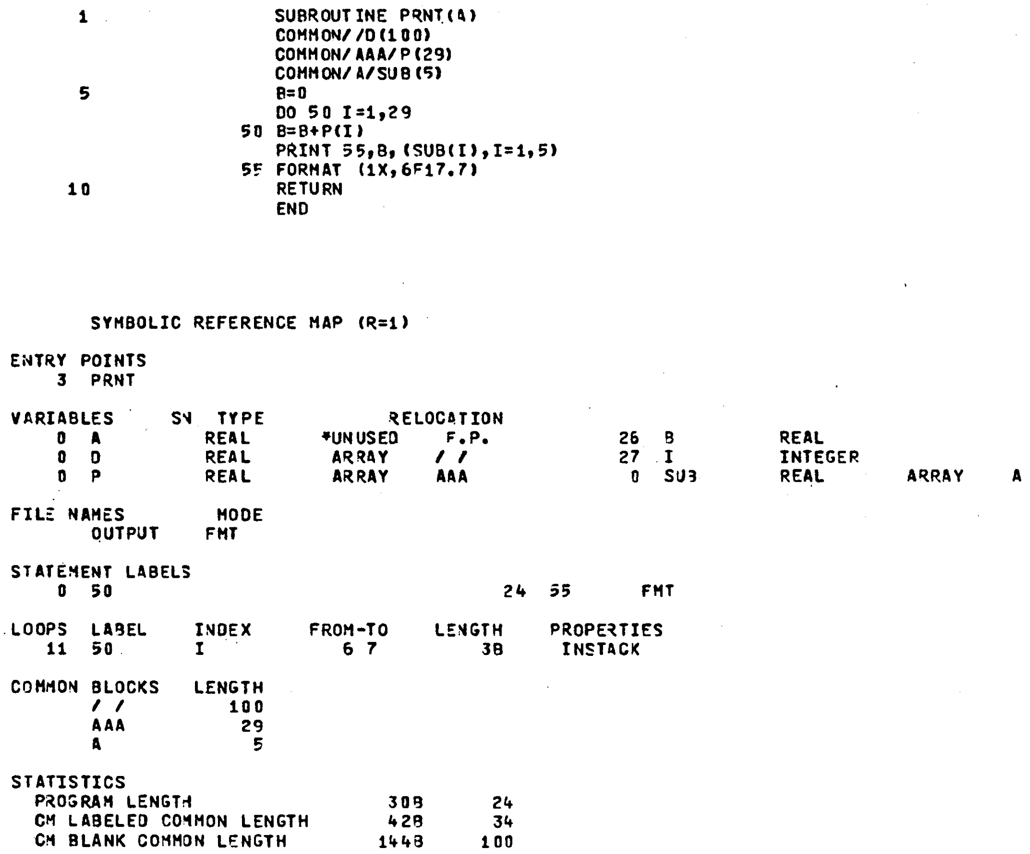
```
        SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
 4107   TESTA                                                     Relative address location of variable B

VARIABLES      SN   TYPE            RELOCATION
    0  A            REAL       ARRAY    AAA            12  B          REAL       ARRAY    AAA    Common block
    8  BLK          REAL       ARRAY    BLOCKA       4166  BLOCKA     REAL                        containing variable B
   24  C            REAL       ARRAY    AAA             0  E          REAL       ARRAY    / /
 4167  I            INTEGER                          4170  J          INTEGER
 4171  N            INTEGER    ARRAY
                                                        Relative address location of variable I
FILE NAMES          MODE
    0  INPUT                      2041  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
 4115  30     * I           16 19        14B                  EXT REFS
 4134  35       J           20 21         4B       INSTACK

COMMON BLOCKS   LENGTH
       / /         100
       AAA          29
       BLOCKA        5

STATISTICS
  PROGRAM LENGTH                154B      108
  BUFFER LENGTH                4103B     2115
  CM LABELED COMMON LENGTH       42B       34
  CM BLANK COMMON LENGTH        144B      100
```

Figure 1-13-2. Main Program of Main Overlay (0,0)



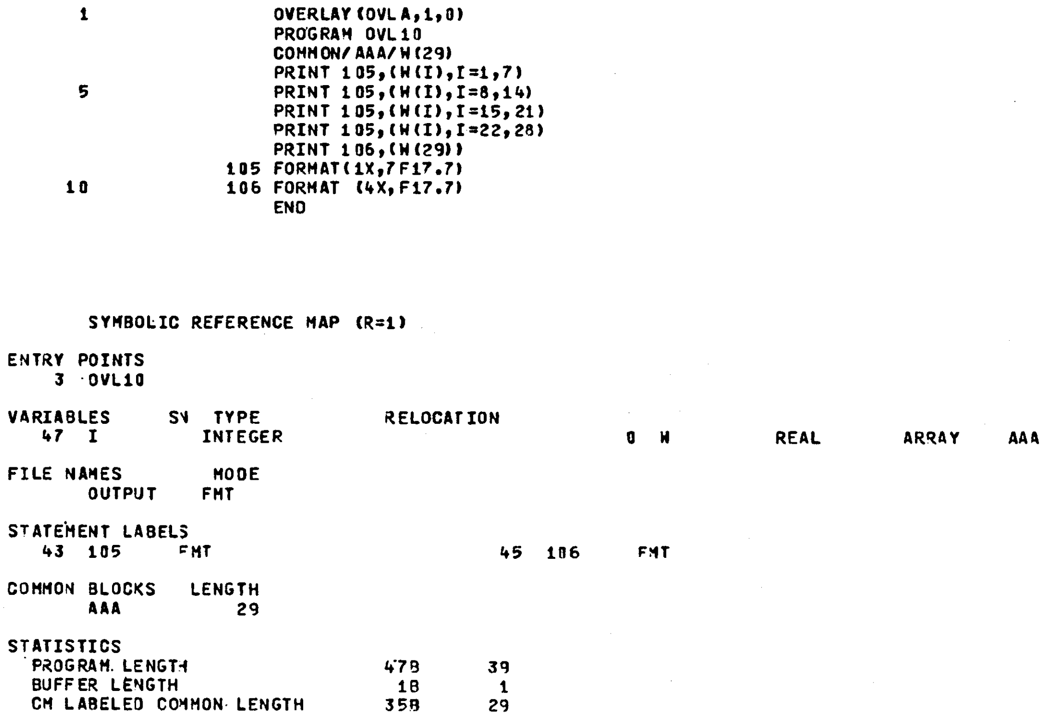
```
      1               FUNCTION TRY(A,B)
                   10 TRY = SQRT(A)+SQRT(B)
                      RETURN
                      ENTRY TRI
      5               IF (A.LE.B) 10,20
                   20 TRY = SQRT(A)-SQRT(B)
                      RETURN
                      END



       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
```

Figure 1-13-3. Function Subroutine of Main Overlay (0, 0)



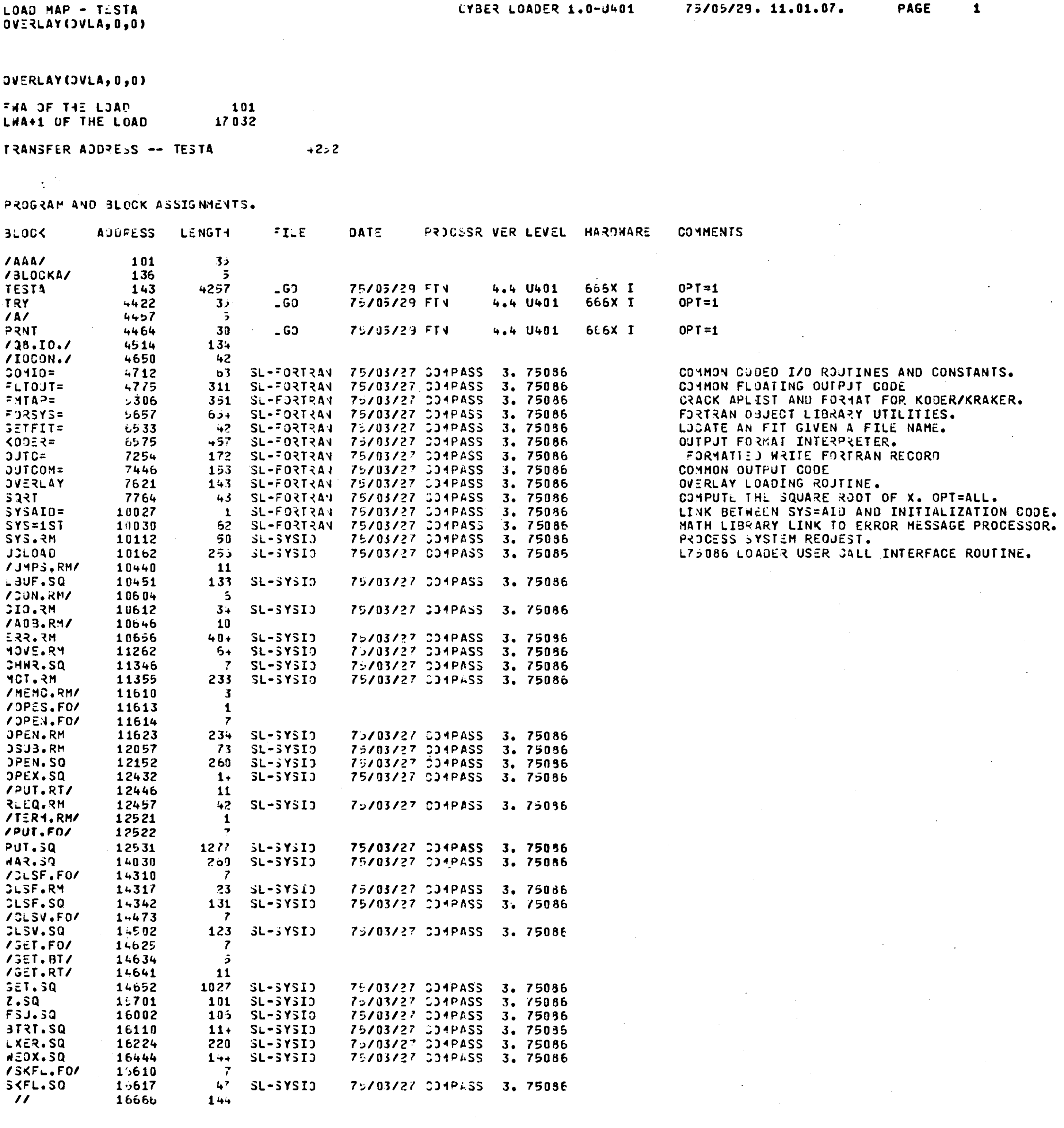
```
      1               SUBROUTINE PRNT(A)
                      COMMON//D(100)
                      COMMON/AAA/P(29)
                      COMMON/A/SUB(5)
      5               B=0
                      DO 50 I=1,29
                   50 B=B+P(I)
                      PRINT 55,B,(SUB(I),I=1,5)
                   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
```

Figure 1-13-4. Subroutine of Main Overlay (0, 0)



```
                OVERLAY(OVLA,1,0)
                PROGRAM OVL10
                COMMON/AAA/W(29)
                PRINT 105,(W(I),I=1,7)
                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)
            106 FORMAT (4X,F17.7)
                END


       SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
    3  OVL10

VARIABLES      SN  TYPE          RELOCATION
   47  I          INTEGER                         0  W         REAL       ARRAY    AAA

FILE NAMES        MODE
       OUTPUT    FMT

STATEMENT LABELS
   43  105      FMT                    45  106      FMT

COMMON BLOCKS   LENGTH
       AAA          29

STATISTICS
  PROGRAM LENGTH               47B       39
  BUFFER LENGTH                 1B        1
  CM LABELED COMMON LENGTH     35B       29
```

Figure 1-13-5. Main Program of Primary Overlay (1,0)



```
LOAD MAP - TESTA                                              CYBER LOADER 1.0-U401      75/05/29. 11.01.07.      PAGE     1
OVERLAY(OVLA,0,0)


OVERLAY(OVLA,0,0)

FWA OF THE LOAD           101
LWA+1 OF THE LOAD       17032

TRANSFER ADDRESS -- TESTA           4252


PROGRAM AND BLOCK ASSIGNMENTS.

BLOCK       ADDRESS   LENGTH     FILE      DATE      PROCSSR VER LEVEL  HARDWARE   COMMENTS

/AAA/          101       35
/BLOCKA/       136        5
TESTA          143     4257      _GO      75/05/29 FTN     4.4 U401    666X I     OPT=1
TRY           4422       35      _GO      75/05/29 FTN     4.4 U401    666X I     OPT=1
/A/           4457        5
PRNT          4464       30      _GO      75/05/29 FTN     4.4 U401    666X I     OPT=1
/Q8.IO./      4514      134
/IOCON./      4650       42
COMIO=        4712       63  SL-FORTRAN  75/03/27 COMPASS  3. 75086             COMMON CODED I/O ROUTINES AND CONSTANTS.
FLTOUT=       4775      311  SL-FORTRAN  75/03/27 COMPASS  3. 75086             COMMON FLOATING OUTPUT CODE
FMTAP=        5306      351  SL-FORTRAN  75/03/27 COMPASS  3. 75086             CRACK APLIST AND FORMAT FOR KODER/KRAKER.
FORSYS=       5657      654  SL-FORTRAN  75/03/27 COMPASS  3. 75086             FORTRAN OBJECT LIBRARY UTILITIES.
GETFIT=       6533       42  SL-FORTRAN  75/03/27 COMPASS  3. 75086             LOCATE AN FIT GIVEN A FILE NAME.
KODER=        6575      457  SL-FORTRAN  75/03/27 COMPASS  3. 75086             OUTPUT FORMAT INTERPRETER.
OUTC=         7254      172  SL-FORTRAN  75/03/27 COMPASS  3. 75086              FORMATTED WRITE FORTRAN RECORD
OUTCOM=       7446      153  SL-FORTRAN  75/03/27 COMPASS  3. 75086             COMMON OUTPUT CODE
OVERLAY       7621      143  SL-FORTRAN  75/03/27 COMPASS  3. 75086             OVERLAY LOADING ROUTINE.
SQRT          7764       43  SL-FORTRAN  75/03/27 COMPASS  3. 75086             COMPUTE THE SQUARE ROOT OF X. OPT=ALL.
SYSAID=      10027        1  SL-FORTRAN  75/03/27 COMPASS  3. 75086             LINK BETWEEN SYS=AID AND INITIALIZATION CODE.
SYS=1ST      10030       62  SL-FORTRAN  75/03/27 COMPASS  3. 75086             MATH LIBRARY LINK TO ERROR MESSAGE PROCESSOR.
SYS.RM       10112       50  SL-SYSIO    75/03/27 COMPASS  3. 75086             PROCESS SYSTEM REQUEST.
OCLOAD       10162      255  SL-SYSIO    75/03/27 COMPASS  3. 75085             L75086 LOADER USER CALL INTERFACE ROUTINE.
/JMPS.RM/    10440       11
LBUF.SQ      10451      133  SL-SYSIO    75/03/27 COMPASS  3. 75086
/CON.RM/     10604        5
CIO.RM       10612       34  SL-SYSIO    75/03/27 COMPASS  3. 75086
/ADB.RM/     10646       10
ERR.RM       10656      404  SL-SYSIO    75/03/27 COMPASS  3. 75086
MOVE.RM      11262       64  SL-SYSIO    75/03/27 COMPASS  3. 75086
CHWR.SQ      11346        7  SL-SYSIO    75/03/27 COMPASS  3. 75086
MCT.RM       11355      233  SL-SYSIO    75/03/27 COMPASS  3. 75086
/MEMC.RM/    11610        3
/OPES.FO/    11613        1
/OPEN.FO/    11614        7
OPEN.RM      11623      234  SL-SYSIO    75/03/27 COMPASS  3. 75086
OSJB.RM      12057       73  SL-SYSIO    75/03/27 COMPASS  3. 75086
OPEN.SQ      12152      260  SL-SYSIO    75/03/27 COMPASS  3. 75086
OPEX.SQ      12432       14  SL-SYSIO    75/03/27 COMPASS  3. 75086
/PUT.RT/     12446       11
RLEQ.RM      12457       42  SL-SYSIO    75/03/27 COMPASS  3. 75086
/TERM.RM/    12521        1
/PUT.FO/     12522        7
PUT.SQ       12531     1277  SL-SYSIO    75/03/27 COMPASS  3. 75086
WAR.SQ       14030      260  SL-SYSIO    75/03/27 COMPASS  3. 75086
/CLSF.FO/    14310        7
CLSF.RM      14317       23  SL-SYSIO    75/03/27 COMPASS  3. 75086
CLSF.SQ      14342      131  SL-SYSIO    75/03/27 COMPASS  3. 75086
/CLSV.FO/    14473        7
CLSV.SQ      14502      123  SL-SYSIO    75/03/27 COMPASS  3. 75086
/GET.FO/     14625        7
/GET.BT/     14634        5
/GET.RT/     14641       11
GET.SQ       14652     1027  SL-SYSIO    75/03/27 COMPASS  3. 75086
Z.SQ         15701      101  SL-SYSIO    75/03/27 COMPASS  3. 75086
FSJ.SQ       16002      105  SL-SYSIO    75/03/27 COMPASS  3. 75086
BTRT.SQ      16110      114  SL-SYSIO    75/03/27 COMPASS  3. 75085
LXER.SQ      16224      220  SL-SYSIO    75/03/27 COMPASS  3. 75086
WEOX.SQ      16444      144  SL-SYSIO    75/03/27 COMPASS  3. 75086
/SKFL.FO/    16610        7
SKFL.SQ      16617       47  SL-SYSIO    75/03/27 COMPASS  3. 75086
 //          16666      144
```

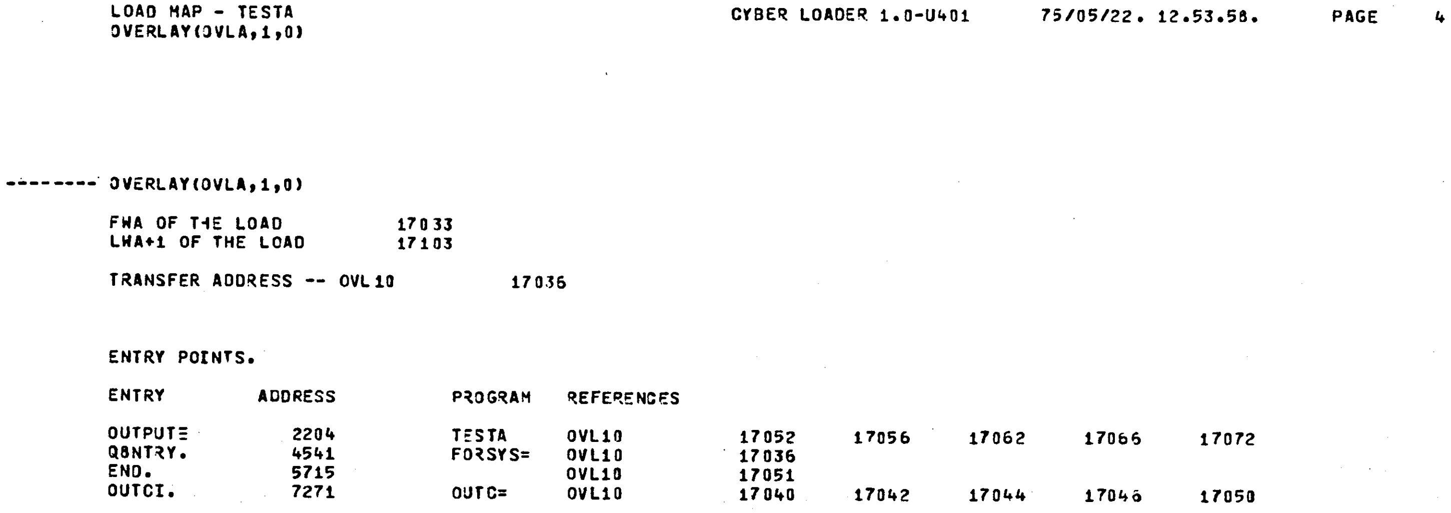
Figure 1-13-6. Loader Map of Main Overlay (0,0)

```
ENTRY POINTS.

ENTRY          ADDRESS          PROGRAM   REFERENCES

OUTPUT=          2204           TESTA     PRNT          4503
TRY              4426           TRY       TESTA         4264
TRI              4436                     TESTA         4270
PRNT             4467           PRNT      TESTA         4256      4306
Q8NTRY.          4541           FORSYS=   TESTA         4252
FECCHR.          4717           COMIO=    KODER=        6653      6740
FECPRT.          4737                     FMTAP=        5575
FEOFAL.          4775           FLTOUT=   KODER=        7035      7054      7075
FEOEOV.          5017                     KODER=        7054
FEOEXP.          5021                     KODER=        7036
FEORND.          5056                     KODER=        7023      7051      7072
FEOSCA.          5113                     KODER=        7013      7046      7066
FEOZRO.          5177                     KODER=        7024      7052      7073
FECNAP.          5316           FMTAP=    KODER=        6660      6735      6741      6761      6776      7236      7245
FECAP.           5324                     OUTC=         7360
FECFMT.          5343                     KODER=        6655      7201      7203      7213      7243
FECFMU.          5346                     KODER=        6614      6720
FECJP=           5451                     OUTC=         7353
FECLP.           5452                     KODER=        6647
FECRP.           5475                     KODER=        6650
FECEE.           5512                     KODER=        7001
FECV.            5560                     KODER=        6624
FECBUG.          5567                     KODER=        6575      6576
END.             5715           FORSYS=   TESTA         4311
IOERR.           6012                     OUTC=         7433
SYSERR.          6114                     COMIO=        4756      4771
                                          FMTAP=        5614
                                          GETFIT=       6562
                                          OUTC=         7440
                                          OVERLAY       7730
FECOPE.          6406                     OUTC=         7335
LINLIM.          6451                     OUTC=         7417
DBGFIT.          6476                     OVERLAY       7662
GETFIT.          6536           GETFIT=   OUTC=         7277
KOJPT.           6576           KOJER=    OUTC=         7267
KODWRT=          7237                     OUTC=         7347
KOREP.           7244                     OUTC=         7270
OUTCI.           7271           OUTC=     PRNT          4501
FEOL.            7446           OJTCOM=   KODER=        6613      6656
FEOI.            7451                     KODER=        6610      6611
FEOXFL.          7516                     FLTOUT=       5020      5210
                                          KODER=        7077      7146
FEOAFM.          7524                     FLTOUT=       5044      5047      5051      5055      5206
FEOBLS.          7531                     FLTOUT=       5000      5001      5002      5004      5174      5175      5201
                                                        5203
                                          KODER=        7030      7155
FEOCNV.          7544                     FLTOUT=       5016
FEOR1F.          7575                     FLTOUT=       5172
OVERLAY          7623           OVERLAY   TESTA         4254      4304
SQRT.           10004           SQRT      TRY           4431      4433      4447      4451
SYSAID=         10027           SYSAID=   Q8.IO.        4540
                                          SYS=1ST      10047
SYS1ST.         10033           SYS=1ST   SQRT         10017
SYS=           |10114           SYS.RM |  Q8.IO.        4536      4546
                                          FORSYS=       5701      5733      5772
                                          ERR.RM       11067     11106     11237
MSG=            10143                     Q8.IO.        4544
                                          FORSYS=       5726      5731      5761      5763      5770      6267      6306
                                                        6314
                                          ERR.RM       11024     11044     11115
                                          PUT.SQ       13262     13316
                                          LXER.SQ      16355
LOADER          10162           UCLOAD    OVERLAY       7704
LBUF.SQ         10451           LBUF.SQ   LXER.SQ      16344
RM.CIO          10613           CIO.RM    ERR.RM       11103
                                          OPEN.SQ      12275     12410
                                          OPEX.SQ      12437
                                          PUT.SQ       12600     13223     13310     13330     13541     13551     13634
                                                       14020
                                          WAR.SQ       14132     14306
                                          CLSF.SQ      14404     14471
                                          CLSV.SQ      14535     14545     14561     14571     14603
                                          GET.SQ       15104     15271     15364     15521     15533
                                          WEOX.SQ      16532     16552
```

Figure 1-13-6. Loader Map of Main Overlay (0, 0) (Contd)

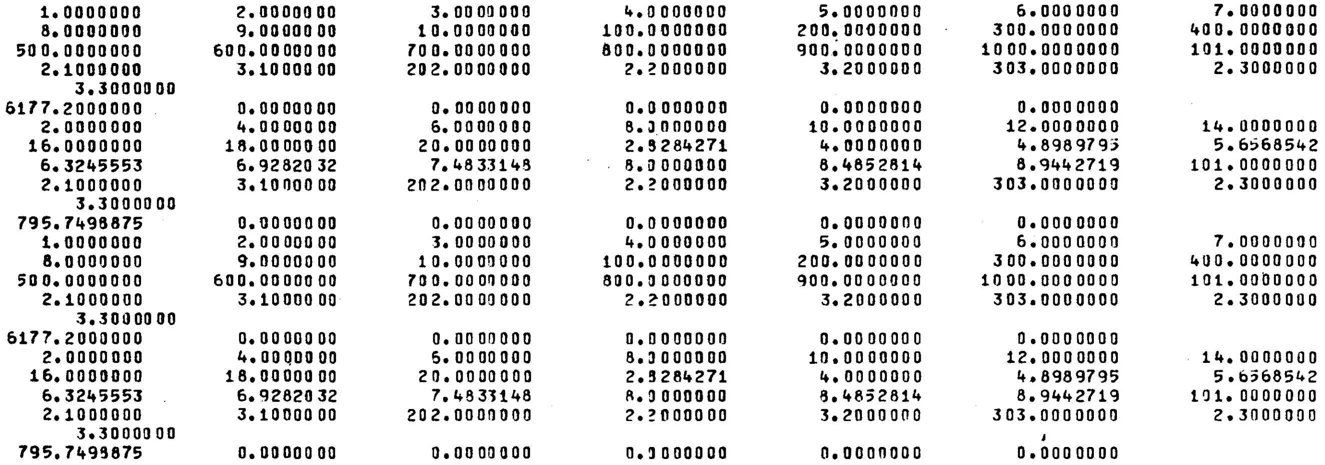
```
RM.RCLA         10623                    ERR.RM      11073
                                         PUT.SQ      13522   13657   14007
                                         WAR.SQ      14232
                                         CLSF.SQ     14372
                                         GET.SQ      14752   15026
                                         BTRT.SQ     16210
RM.RCLP         10630                    PUT.SQ      13236   13256
                                         GET.SQ      15245
ERR.RM          10734          ERR.RM    JMPS.RM     10441   10442   10443   10444   10445   10446   10450
                                         MEMC.RM     11610   11611   11612
                                         OPES.FO     11613
                                         OPEN.FO     11614   11615   11616   11617   11620   11621   11622
                                         OPEN.RM     11625   11716   11765   12031   12040   12042   12046
                                                     12050   12051
                                         OSUB.RM     12140   12141   12142   12143   12144   12145   12146
                                                     12147   12150
                                         OPEN.SQ     12226   12230   12233   12274   12334   12336   12373
                                                     12423   12424
                                         OPEX.SQ     12433
                                         RLEQ.RM     12462
                                         PUT.FO      12522   12523   12524   12525   12526   12527   12530
                                         PUT.SQ      12531   12536   12572   12613   12616   12623   12672
                                                     12734   12770   13017   13225   13254   13255   13275
                                                     14015   14023
                                         CLSF.FO     14310   14311   14312   14313   14314   14315   14316
                                         CLSF.RM     14321   14332   14340
                                         CLSF.SQ     14350   14427   14472
                                         CLSV.FO     14473   14474   14475   14476   14477   14500   14501
                                         GET.FO      14625   14626   14627   14630   14631   14632   14633
                                         GET.SQ      14652   14656   14673   14700   14705   15405   15527
                                                     15562   15605
                                         LXER.SQ     16371
                                         WEOX.SQ     16454   16464   16467   16500
                                         SKFL.FO     16610   16611   16612   16613   16614   16615   16616
                                         SKFL.SQ     16621   16624   16627
MOVE.RM         11262          MOVE.RM   PUT.SQ      13111
                                         FSU.SQ      16061
CHWR.SQ         11346          CHWR.SQ   OPEN.SQ     12162
                                         PUT.SQ      13026
MCT.RM          11362          MCT.RM    OPEN.RM     11646   11650   11752   11754   12022   12026
                                         CLSF.RM     14324
OPEN.RM         11623          OPEN.RM   FORSYS=      6447
OSUB.RM         12057          OSJB.RM   OPEN.RM     11766
OPEN.SQ         12152          OPEN.SQ   FORSYS=      6447
OPXX.SQ         12267                    OPEX.SQ     12445
OPEX.SQ         12432          OPEX.SQ   OPEN.SQ     12266
RLEQ.RM         12457          RLEQ.RM   PUT.SQ      12644   12726
                                         WAR.SQ      14031
PUT.SQ          12532          PJT.SQ    FORSYS=      6331
                                         OUTC=        7423
WAR.SQ          14030          WAR.SQ    PUT.SQ      12622
                                         WEOX.SQ     16477
CLSF.RM         14317          CLSF.RM   FORSYS=      6101    6140
                                         OUTC=        7327
RSPT.SQ         14430          CLSF.SQ   OPEN.SQ     12345
                                         OPEX.SQ     12434
                                         CLSV.SQ     14521   14615
                                         GET.SQ      15027   15513
CLSV.SQ         14504          CLSV.SQ   PUT.SQ      13300   13532
                                         GET.SQ      15622
SKGT.SQ         14722          GET.SQ    SKFL.SQ     16645   16653
GXIT.SQ         15030                    Z.SQ        15730   15765   15777
                                         FSU.SQ      16036   16052   16075   16077
GRTJ.SQ         15111                    FSU.SQ      16037   16102
ANBL.SQ         15150                    BTRT.SQ     16122   16140   16145   16223
AMAC.SQ         15154                    FSU.SQ      16040   16102   16107
                                         BTRT.SQ     16110   16123   16147   16150   16157   16160   16161
                                                     16167   16175   16200   16206
DXIT.SQ         15654                    SKFL.SQ     16635
GET.Z           15701          Z.SQ      FORSYS=      6074
RMUD.SQ         16021          FSU.SQ    GET.SQ      15022
RMU2.SQ         16041                    GET.SQ      15016
PUT.C           16110          BTRT.SQ   Q8.IO.       4514
LAB1.SQ         16231          LXER.SQ   Q8.IO.       4514
WEOS.SQ         16451          WEOX.SQ   OVERLAY      7672
SKFL.SQ         16617          SKFL.SQ   FORSYS=      6074
```

Figure 1-13-6. Loader Map of Main Overlay (0, 0) (Contd)



```
LOAD MAP - TESTA                                  CYBER LOADER 1.0-U401    75/05/22. 12.53.58.    PAGE    4
OVERLAY(OVLA,1,0)



-------- OVERLAY(OVLA,1,0)

         FWA OF THE LOAD          17033
         LWA+1 OF THE LOAD        17103

         TRANSFER ADDRESS -- OVL10          17036


         ENTRY POINTS.

         ENTRY        ADDRESS          PROGRAM    REFERENCES

         OUTPUT=        2204           TESTA      OVL10        17052    17056    17062    17066    17072
         Q8NTRY.        4541           FORSYS=    OVL10        17036
         END.           5715                      OVL10        17051
         OUTCI.         7271           OUTC=      OVL10        17040    17042    17044    17046    17050
```

Figure 1-13-7. 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 1-13-8. Program Output

```
EXCHANGE PACKAGE.
 P    10114  A0    2204  B0       0        (A0)  1725 2420 2524 0000 0000
 RA  152500  A1       1  B1       1        (A1)  0516 0420 0000 0000 0000
 FL   17200  A2    6530  B2  777755        (A2)  1717 0631 4631 4640 3615
 EM       7  A3    6531  B3    2450        (A3)  2000 0000 0000 0000 0012
 RAX      0  A4    5670  B4      24        (A4)  5555 5555 5555 5733 3637
 FLX      0  A5    5674  B5   10444        (A5)  2000 0000 0000 0003 1561
 MA    1400  A6       1  B6    6102        (A6)  0516 0420 0000 0000 0000
             A7    2236  B7      30        (A7)  0000 0000 0000 0000 0000

 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 0003
 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


DUMP FROM  10054 TO 10154
  10054     62577 77676 66622 53455     06600 10056 54355 55431     03410 10057 56330 56431     03610 10060 56330 57431
  10060     10633 55341 22704 55431     76165 51600 10103 55761     10633 74260 22704 55671     21111 55761 55071 46000
  10064     01000 00000 61000 46000     51300 10076 61100 00001     43002 55231 26050 53735     21322 55121 63635 55411
  10070     21322 53040 04000 10030     00000 00000 00000 00000     00000 00000 00000 00000     00000 00000 00000 00000
  10074     00000 00000 00000 00000     00000 00000 00000 00000     00000 00000 00000 00000     11160 61116 11240 50000
  10100     55012 20725 15051 62455     11160 40506 11161 12405     00000 00000 00000 00000     00000 00000 00000 00000
  10104     00000 00000 00000 00000     00000 00000 00000 00000     00000 00000 00000 00000     00000 00000 00000 10074
  10110     00000 00000 00000 00000     00000 00000 00000 00000     04000 10125 00000 00000     01300 00000 00000 00000
  10114     04000 05734 00000 00000     51100 00001 03110 10115     54610 04000 10113 46000     51100 00066 03310 10121
  10120     51100 10112 04000 10122     71100 00130 20160 46000     13661 13161 13661 46000     51600 10113 10611 46000
  10124     51100 00001 01000 10112     20652 01000 10114 46000     51100 00001 03110 10126     04004 10127 61000 46000
  10130     51100 00001 03110 10127     71502 20314 04000 10125     20150 36661 01000 10114     04004 10133 61000 46000
  10134     71602 20314 20652 36662     53160 20173 03310 10133     03010 10133 51100 00001     03110 10135 71100 00001
  10140     04000 10132 61000 46000     71603 24616 12661 20651     01000 10114 61000 46000     04000 05732 00000 00000
  10144     20622 12161 20651 20123     03260 10141 71600 00301     20645 13116 04000 10141     01000 10133 61000 46000
  10150     43652 71100 00002 12661     53120 11161 71600 31117     03270 10153 14777 27606     12717 20652 53720 12662
  10154     73220 01000 10114 46000     04004 10155 61000 46000     53120 20173 03310 10150     03110 10147 52120 00001
```

Figure 1-13-9. Exchange Package Dump

```
DUMP FROM      0 TO 10000
      0     00000 00000 00000 00000     05160 42000 00000 00000     11162 02524 00000 00143     17252 42025 24000 02204
      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 17200 00000 00001     00000 00000 00000 00000     00000 00000 00000 00000
     64     17261 40100 00000 00000     40000 00000 00000 17103     40242 00000 01000 17032     40000 00000 40000 00000
     70     17261 40157 55000 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     00000 00001 71030 04252/AAA(A)17214 00000 00000 00000     17224 00000 00000 00000     17226 00000 00000 00000
    104     17234 00000 00000 00000     17235 00000 00000 00000     17236 00000 00000 00000     17237 00000 00000 00000
    110     17244 00000 00000 00000     17244 40000 00000 00000     17245 00000 00000 00000 (B) 17215 52023 63147 74736
    114     17224 00000 00000 00000     17224 71421 60500 44411     17225 52023 53147 74736     17226 24613 01655 51333
    120     17226 73317 27205 41145     17227 36735 20426 40772     17234 00000 00000 00000     17234 17416 66315 75547
    124     17234 36156 74671 37646 (C) 17226 00000 00000 00000 /BLOCK A/ (BLK) 17214 14631 46314 63146     17216 14631 46314 63146
    130     17276 24000 00000 00000     17214 31463 14631 46315     17216 31463 14631 46315     17304 57000 00000 00000
    134     17214 46314 63146 31463     17216 46314 63146 31463     17226 00000 00000 00000     17236 00000 00000 00000
    140     17244 40000 00000 00000     17246 00000 00000 00000     17247 40000 00000 00000     11162 02524 00000 00000
    144     00000 00000 00000 00166     00000 00042 60000 00000     00000 00000 20010 00000     00000 00000 00000 00000
    150     00000 00002 00000 00203     00000 00000 01400 00000     00000 00000 00000 00000     00000 00000 00000 00000
    154     00000 00000 00000 00000     00000 00000 00000 00000     00000 00000 00000 00000     00000 00000 00000 00000
        DUPLICATED LINES.
   2204     17252 42025 24000 00000     00000 14700 40630 02227     00000 00042 50010 00000     00000 00000 20010 00000
   2210     00000 00000 00000 06103     00000 00003 00000 02244     00000 22600 03400 04521     00000 00000 00000 00000
   2214     00000 00000 00000 00000     00000 00000 00000 00000     00000 00002 00000 00000     00000 00002 00000 00000
   2220     00000 10000 00000 00000     00300 00000 00000 02244     00000 00000 00000 00000     00000 00000 00000 02244
   2224     00000 00000 00002 24400     00000 00000 00000 00000     00000 00000 00000 00000     17252 42025 24000 00131
   2230     04111 46000 00040 02244     00000 00000 00000 02244     00000 00000 00000 02244     37260 00000 01000 04245
   2234     00000 00000 00000 00000     00000 02237 00000 00000     00000 00000 00000 00000     00000 00000 00000 00001
   2240     00000 00000 00000 00000     00000 00000 00000 00000     00000 00000 00000 00000     00000 00000 00000 00000
   2244     55555 55555 55555 55534     57333 33333 33333 35555     55555 55555 55355 73333     33333 33333 55555 55555
   2250     55555 53657 33333 33333     33335 55555 55555 55555     37573 33333 33333 33355     55555 55555 55554 05733
   2254     33333 33333 33555 55555     55555 55541 57333 33333     33333 35555 55555 55555     55425 73333 33333 33333
   2260     00000 00000 00000 00000     55555 55555 55555 55543     57333 33333 33333 35555     55555 55555 55445 73333
   2264     33333 33333 55555 55555     55553 43357 33333 33333     33335 55555 55555 53433     33573 33333 33333 33355
   2270     55555 55555 35333 35733     33333 33333 33555 55555     55553 63333 57333 33333     33333 35555 55555 55537
   2274     33335 73333 33333 33333     00000 00000 00000 00000     55555 55555 55554 03333     57333 33333 33333 35555
   2300     55555 55541 33335 73333     33333 33333 55555 55555     55423 33357 33333 33333     33335 55555 55555 54333
   2304     33573 33333 33333 33355     55555 55555 44333 35733     33333 33333 33555 55555     55343 33333 57333 33333
   2310     33333 35555 55555 55534     33345 73333 33333 33333     00000 00000 00000 00000     55555 55555 55555 55535
   2314     57343 33333 33333 35555     55555 55555 55365 73433     33333 33333 55555 55555     55353 33557 33333 33333
   2320     33335 55555 55555 55555     35573 53333 33333 33355     55555 55555 55553 65735     33333 33333 33555 55555
   2324     55553 63336 57333 33333     33333 35555 55555 55555     55355 73633 33333 33333     00000 00000 00000 00000
   2330     55555 55555 55555 55555     55553 65736 33333 33333     33000 00000 00000 00000     55555 55555 55413 44242
   2334     57353 33333 33333 35555     55555 55555 55335 73333     33333 33333 55555 55555     55555 53357 33333 33333
   2340     33335 55555 55555 55555     33573 33333 33333 33355     55555 55555 55553 35733     33333 33333 33555 55555
   2344     55555 55533 57333 33333     33333 30000 00000 00000     55555 55555 55555 55535     57333 33333 33333 35555
   2350     55555 55555 55375 73333     33333 33333 55555 55555     55555 54157 33333 33333     33335 55555 55555 55555
   2354     43573 33333 33333 33355     55555 55555 55343 35733     33333 33333 33555 55555     55555 53435 57333 33333
```

Figure 1-13-10. Central Memory Dump

```
2360        33333 35555 55555 55555     34375 73333 33333 33333     00000 00000 00000 00000     55555 55555 55555 53441
2364        57333 33333 33333 35555     55555 55555 34435 73333     33333 33333 55555 55555     55553 53357 33333 33333
2370        33335 55555 55555 55555     35574 33543 37354 23455     55555 55555 55553 75733     33333 33333 33555 55555
2374        55555 55537 57434 44344     42444 05555 55555 55555     55405 74140 41434 03735     00000 00000 00000 00000
2400        55555 55355 55555 55541     57363 53740 40403 55555     55555 55555 55415 74435     43353 33635 55555 55555
2404        55555 54257 37433 63634     37435 55555 55555 55555     43573 33333 33333 33355     55555 55555 55554 35737
2410        43403 54334 37555 55555     55555 55543 57443 73735     42344 45555 55555 55534     33345 73333 33333 33333
2414        00000 00000 00000 00000     55555 55555 55555 55535     57343 33333 33333 35555     55555 55555 55365 73433
2420        33333 33333 55555 55555     55353 33557 33333 33333     33335 55555 55555 55555     35573 53333 33333 33355
2424        55555 55555 55553 65735     33333 33333 33555 55555     55553 63336 57333 33333     33333 35555 55555 55555
2430        55355 73633 33333 33333     00000 00000 00000 00000     55555 55555 55555 55555     55553 65736 33333 33333
2434        33000 00000 00000 00000     55555 55555 55554 24440     57423 74443 43424 05555     55555 55555 55335 73333
2440        33333 33333 55555 55555     55555 53357 33333 33333     33335 55555 55555 55555     33573 33333 33333 33355
2444        55555 55555 55553 35733     33333 33333 33555 55555     55555 55533 57333 33333     33333 30000 00000 00000
2450        00000 00000 00000 00000     00000 00000 00000 00000     00000 00000 00000 00000     00000 00000 00000 00000
      DUPLICATED LINES.
4244        00000 00000 00000 00000     00000 22600 00000 04521     00000 22600 00000 04521     77777 77777 77777 66167
4250        24052 32401 55550 04252     00000 00000 00000 00000     51100 04245 01000 04541     51100 04312 46000 46000
4254        01000 07623 00160 04250     51100 04316 45000 46000     01000 04467 00170 04250     71700 00001 51700 04332
4260        51500 04332 52450 00100     72750 00100 40644 46000     51700 04320 51700 04321     54640 51100 04320 46000
4264        01000 04426 00220 04250     51500 04332 72750 00100     52650 00100 51700 04320     51700 04321 51100 04320
4270        01000 04436 00230 04250     51500 04332 72750 00001     72077 77764 52650 00112     54750 03300 04260 46000
4274        71700 00001 51700 04333     51500 04333 61600 00005     36055 63750 62500 00100     51570 00100 56450 30045
4300        61550 00002 24700 46000     51770 00135 61770 00001     06670 04277 76770 46000     51700 04333 51100 04323
4304        01000 07623 00260 04250     51100 04316 46000 46000     01000 04467 00270 04250     51500 04333 51100 04250
4310        27005 24700 51700 00125     04000 05715 46000 46000     00000 00000 00000 04416     00000 00000 00000 04327
4314        00000 00000 00000 04330     00000 00000 00000 00000     00000 00000 00000 04331     00000 00000 00000 00000
4320        00000 00000 00000 00112     00000 00000 00000 00112     00000 00000 00000 00000     00000 00000 00000 04420
4324        00000 00000 00000 04327     00000 00000 00000 04330     00000 00000 00000 00000     00000 00000 00000 00001
4330        00000 00000 00000 00000     00000 00000 00000 00000    |00000 00000 00000 00013| I  00000 00000 00000 00006
4334        00000 00000 00000 00173     00000 00000 00000 00173     00000 00000 00000 00173     00000 00000 00000 00173
      DUPLICATED LINES.
4414        00000 00000 00000 00173     00000 00000 00000 00173     17261 40155 55555 55555     00000 00000 00000 00000
4420        17261 40155 55555 55555     00000 00000 00000 00000     24223 15555 55550 04426     00000 00000 00000 17200
4424        51400 04456 10644 46000     51300 04423 52030 00000     04000 04271 00000 00000     74600 54010 51600 04423
4430        46000 46000 46000 46000     54500 53150 01000 10004     50500 00001 53150 46000     51600 04454 01000 10004
4434        51500 04454 30056 24700     51700 04456 04000 04424     04000 04271 00000 00000     51200 04441 10622 46000
4440        51600 04430 04000 04427     04000 04442 61000 46000     51100 04455 51200 04436     10611 22702 51600 04430
4444        51700 04426 61000 46000     54500 50400 00001 53150     53340 31031 24700 46000     03270 04431 01000 10004
4450        50500 00001 53150 46000     51600 04454 01000 10004     51500 04454 31056 24700     51700 04456 04000 04424
4454        17224 36156 74671 37646     46000 46000 45000 46000     17234 36156 74671 37646     00000 00000 00000 00000
4460        00000 00000 00000 00000     00000 00000 00000 00000     00000 00000 00000 00000     00000 00000 00000 00000
4464        20221 62455 55550 04467     00000 00000 00000 17200     51300 04465 52030 00000     04000 04307 00000 00000
4470        74600 54010 51600 04465     43700 71600 00001 46000     51700 04512 51600 04513     51500 04513 61600 00035
4474        63750 46000 46000 46000     51500 04512 51470 00100     61770 00001 30045 24700     54750 06670 04475 76770
4500        51700 04513 51100 04503     01000 07271 00100 04464     04000 04466 46000 46000     00000 00000 00000 02204
4504        00000 00000 00000 04510     00030 00000 01000 04512     00030 00000 05000 04457     00000 00000 00000 00000
4510        55404 05555 55000 00000     51343 04106 34425 74252     17316 15677 76120 00744     00000 00000 00000 00036
4514        00000 00000 00000 04245     00000 00000 00000 00000     03172 03122 11071 02455     00000 00000 00000 00000
4520        00000 00000 00000 00000     55555 55555 55554 24440     57423 74443 43424 05555     55555 55555 55335 73333
4524        33333 33333 55555 55555     55555 53357 33333 33333     33335 55555 55555 55555     33573 33333 33333 33355
4530        55555 55555 55553 35733     33333 33333 33555 55555     55555 55533 57333 33333     33333 33333 33333 33333
4534        55355 73633 33333 33333     20652 12661 43101 20151     12661 01000 10114 46000     51200 04531 10722 46000
4540        00000 00000 00000 04506     00000 00000 74410 04316     00000 00000 00000 02204     00000 00000 00000 04504
4544        00000 00000 00000 00000     00000 00000 00000 00000     00000 00000 00000 00006     77777 77777 77777 77776
4550        00000 00000 00000 00000     00000 00000 00000 00007     00000 00000 00000 00000     00000 00000 00000 00001
4554        00000 00000 00000 00004     00000 00000 00000 00000     00000 00000 00000 00226     20060 00000 00000 00021
4560        11701 54335 06020 04576     51200 04626 11403 37623     00000 00000 00000 00002     00000 00000 00000 00000
4564        20120 00001 01000 04512     00000 00000 00000 00000     00000 00000 00000 04521     04000 07244 61000 45000
4570        00000 00000 00000 00057     51600 04627 66710 54300     01000 04640 61000 46000     51500 04627 61500 00001
4574        20120 00000 00000 04511     43052 04000 04623 46000     67225 03310 04623 53210     03120 04600 53710 46000
4600        10255 52510 00005 46000     71600 00001 43773 20535     15667 11775 36667 20637     54650 52510 00006 46000
4604        71600 00006 43771 20546     15667 11775 36667 20626     54650 52510 00002 46000     71600 00002 43772 20532
4610        15667 11775 36667 20642     54650 52510 00002 46000     71600 00002 43772 20536     15667 11775 36667 20636
4614        54650 52510 00006 43752     15457 52510 00006 43744     21544 15657 61500 00001     03140 04621 71600 00226
4620        71400 04521 61000 46000     20544 12764 54710 10611     22502 54640 54115 54445     03210 04560 07020 04560
4624        03350 04532 14155 46000     04000 04532 61000 46000     20140 00000 00000 00002     00000 00000 00000 00000
4630        22227 36727 20603 15213     20701 12661 36771 53525     20506 15150 63410 22244     73113 11505 03320 04630
4634        66357 04400 04637 66322     03150 04640 04430 04637     67545 05520 04640 66300     43400 04300 04640 10677
4640        00000 00000 61000 46000     71400 07774 43065 76600     66211 13777 61307 77744     20425 66500 71300 00007
4644        04000 04632 61000 46000     00000 00000 00000 00000     00000 00000 00000 00000     00000 00000 00000 00000
4650        55555 55555 55555 55555     77777 77777 77777 77777     77777 77777 77777 77700     77777 77777 77777 70000
4654        77777 77777 77770 00000     77777 77777 77000 00000     77777 77777 00000 00000     77777 77700 00000 00000
4660        77777 70000 00000 00000     77770 00000 00000 00000     77000 00000 00000 00000     00000 00000 00000 00000
4664        17204 00000 00000 00000     17235 00000 00000 00000     17266 20000 00000 00000     17317 64000 00000 00000
4670        17354 70400 00000 00000     17406 06500 00000 00000     17437 50220 00000 00000     17474 61132 00000 00000
4674        17525 75360 40000 00000     17557 34654 50000 00000     17614 52013 71000 00000     17645 64416 67200 00000
4700        17677 21522 45040 00000     17734 43023 47124 00000     17765 53630 40751 00000     20027 06576 51143 20000
4704        20064 34157 11576 00000     20115 43212 74135 42400     20146 74055 53164 73100     20204 25434 43011 04750
```

Figure 1-13-10. Central Memory Dump (Contd)

SYSTEM UTILITY CONTROL STATEMENTS 14

NOS provides the following utilities for file maintenance.

EDIT — Performs data manipulations on a specified mass storage file

MODIFY — Edits a Modify-formatted program library file

OPLEDIT — Removes modification decks and identifiers from a Modify-formatted program library file

UPDATE — Edits an Update-formatted program library file

UPMOD — Converts an Update-formatted program library file to a Modify-formatted program library file

KRONREF — Generates a cross-reference listing of system symbols

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.

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 can be entered only if the X or Q options 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, . . . , 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	Directives
004	All other input statements
010	Modifications made
020	Directives processed from the program library
040	Deck status
100	Directory lists
200	Inactive statements
400	Active statements

If this option is omitted, x=177 is assumed (that is, the first seven 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.

For a complete description of the OPLEDIT utility, refer to the MODIFY Reference Manual.

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) will be a sequential file. If W is omitted, the new program library will be 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.

Z The input file (refer to I option) is assumed to be in PCS-compressed format. This parameter applies to the directives input file only; it does not apply to files specified by READ directives. If Z is omitted, the input file is a normal, coded file.

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 +-*/$=. Note, however, that 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.

Note that 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.

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 SYSTEXT 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

NOS TIME-SHARING 64-CHARACTER SET

The character sets for ALGOL and COBOL are listed in their respective reference manuals.

ASCII CODE TERMINAL†				CORRESPONDENCE CODE TERMINAL††				INTERNAL DISPLAY CODE (6/12-BIT OCTAL)
STANDARD PRINT		APL PRINT		STANDARD PRINT		APL PRINT		
CHAR.	CODE (8-BIT OCTAL)	CHAR.	CODE (8-BIT OCTAL)	CHAR.	CODE (7-BIT OCTAL)	CHAR.	CODE (7-BIT OCTAL)	
:	072	:	276	:	153	:	121	00†††
A	101	A	341	A	171	A	171	01
B	102	B	342	B	166	B	166	02
C	303	C	143	C	172	C	172	03
D	104	D	344	D	052	D	052	04
E	305	E	145	E	112	E	112	05
F	306	F	146	F	163	F	163	06
G	107	G	347	G	043	G	043	07
H	110	H	350	H	046	H	046	10
I	311	I	151	I	031	I	031	11
J	312	J	152	J	103	J	103	12
K	113	K	353	K	032	K	032	13
L	314	L	154	L	106	L	106	14
M	115	M	355	M	141	M	141	15
N	116	N	356	N	122	N	122	16
O	317	O	157	O	105	O	105	17
P	120	P	360	P	013	P	013	20
Q	321	Q	161	Q	133	Q	133	21
R	322	R	162	R	051	R	051	22
S	123	S	363	S	045	S	045	23
T	324	T	164	T	002	T	002	24
U	125	U	365	U	062	U	062	25
V	126	V	366	V	061	V	061	26
W	327	W	167	W	165	W	165	27
X	330	X	170	X	142	X	142	30
Y	131	Y	371	Y	147	Y	147	31
Z	132	Z	372	Z	124	Z	124	32
0	060	0	060	0	144	0	144	33
1	261	1	261	1	040	1	040	34
2	262	2	262	2	020	2	020	35
3	063	3	063	3	160	3	160	36
4	264	4	264	4	004	4	004	37
5	065	5	065	5	010	5	010	40
6	066	6	066	6	130	6	130	41
7	267	7	267	7	150	7	150	42
8	270	8	270	8	070	8	070	43
9	071	9	071	9	064	9	064	44
+	053	+	055	+	023	+	067	45
−	055	−	275	−	067	−	067	46
*	252	*	120	*	070	*	013	47
/	257	/	257	/	007	/	007	50
(	050	(	053	(	064	(	153	51
)	251	)	252	)	144	)	111	52
$	044	$	374	$	004	α	171	53
=	275	=	245	=	023	=	010	54

3AE4A

† THE OCTAL CODES LISTED FOR ASCII CODE TERMINALS ARE SHOWN WITH EVEN PARITY (NORMAL)

†† THE OCTAL CODES LISTED FOR CORRESPONDENCE CODE TERMINALS ARE SHOWN WITH ODD PARITY (NORMAL)

††† USE OF THE COLON IN PROGRAM AND DATA FILES WILL CAUSE PROBLEMS. THIS IS PARTICULARLY TRUE WHEN IT IS USED IN PRINT AND FORMAT STATEMENTS.

ASCII CODE TERMINAL				CORRESPONDENCE CODE TERMINAL				INTERNAL DISPLAY CODE (6/12-BIT OCTAL)
STANDARD PRINT		APL PRINT		STANDARD PRINT		APL PRINT		
CHAR.	CODE (8-BIT OCTAL)	CHAR.	CODE (8-BIT OCTAL)	CHAR.	CODE (7-BIT OCTAL)	CHAR.	CODE (7-BIT OCTAL)	
(SPACE)	240	(SPACE)	240	(SPACE)	100	(SPACE)	100	55
,	254	,	254	,	073	,	073	56
.	056	.	056	.	121	.	121	57
#	243	¨	041	#	160	¨	040	60
[	333	[	273	¼	001	[	153	61
]	335	]	072	½	001	]	111	62
%	245	÷	176	%	010	÷	023	63
"	042	≠	050	¨	111	≠	070	64
_	137†	_	306	_	067	_	163	65
!	041	∨	251	¢	130	∨	064	66
&	246	∧	137	&	150	∧	144	67
'	047	'	113	'	111	'	032	70
?	077	?	321	?	007	?	133	71
<	074	<	243	NULL	——	<	160	72
>	276	>	047	NULL	——	>	150	73
@	300	≤	044	@	020	≤	004	74
\	134	\	077	NULL	——	\	007	75
^	336	¯	042	NULL	——	¯	020	76
;	273	;	074	;	153	;	073	77
NULL	——	◊	134	NULL	——	NULL	——	7600
a	341	α	101	a	171	α	171	7601
b	342	⊥	102	b	166	⊥	166	7602
c	143	∩	303	c	172	∩	172	7603
d	344	⌊	104	d	052	⌊	052	7604
e	145	∊	305	e	112	∊	112	7605
f	146	×	336	f	163	×	023	7606
g	347	∇	107	g	043	∇	043	7607
h	350	Δ	110	h	046	Δ	046	7610
i	151	⍳	311	i	031	⍳	031	7611
j	152	∘	312	j	103	∘	103	7612
k	353	⊣	173	k	032	NULL	——	7613
l	154	⎕	314	l	106	⎕	106	7614
m	355	|	115	m	141	|	141	7615
n	356	⊤	116	n	122	⊤	122	7616
o	157	○	317	o	105	○	105	7617
p	360	←	100	p	013	←	001	7620
q	161	→	140	q	133	→	101	7621
r	162	⍴	322	r	051	⍴	051	7622
s	363	⌈	123	s	045	⌈	045	7623
t	164	~	324	t	002	~	002	7624
u	365	↓	125	u	062	↓	062	7625
v	366	∪	126	v	061	∪	061	7626
w	167	⍵	327	w	165	⍵	165	7627
x	170	⊃	330	x	142	⊃	142	7630
y	371	↑	131	y	147	↑	147	7631

3AE3A

† ON TTY MODELS HAVING NO UNDERLINE, THE BACKARROW (←) TAKES ITS PLACE

ASCII CODE TERMINAL				CORRESPONDENCE CODE TERMINAL				INTERNAL DISPLAY CODE (6/12-BIT OCTAL)
STANDARD PRINT		APL PRINT		STANDARD PRINT		APL PRINT		
CHAR.	CODE (8-BIT OCTAL)	CHAR.	CODE (8-BIT OCTAL)	CHAR.	CODE (8-BIT OCTAL)	CHAR.	CODE (8-BIT OCTAL)	
Z	372	⊂	132	Z	124	⊂	124	7632
{	173	{	335	NULL	——	NULL	——	7633
¦	174	≥	246	±	040	≥	130	7634
}	175	}	175	NULL	——	NULL	——	7635
~	176	⊢	133	NULL	——	NULL	——	7636
DEL	377	DEL	377	NULL	——	NULL	——	7637
NUL	000	NUL	000	NUL	075	NUL	075	7640
SOH	201	SOH	201	SOA	166	SOA	166	7641
STX	202	STX	202	EOA	064	EOA	064	7642
ETX	003	ETX	003	NULL	——	NULL	——	7643
EOT	204	EOT	204	EOT	174	EOT	174	7644
ENQ	005	ENQ	005	NULL	——	NULL	——	7645
ACK	006	ACK	006	ACK	067	NULL	——	7646
BELL	207	BELL	207	NULL	——	NULL	——	7647
BS	210	BS	210	BS	135	BS	135	7650
HT	011	HT	011	HT	057	HT	057	7651
LF	012	LF	012	LF	156	LF	156	7652
VT	213	VT	213	NULL	——	NULL	——	7653
FF	014	FF	014	NULL	——	NULL	——	7654
CR	215	CR	215	CR	155	CR	155	7655
SO	216	SO	216	UCS	034	UCS	034	7656
SI	017	SI	017	LCS	037	LCS	037	7657
DLE	220	DLE	220	NULL	——	NULL	——	7660
DC1	021	DC1	021	NULL	——	NULL	——	7661
DC2	022	DC2	022	NULL	——	NULL	——	7662
DC3	023	DC3	023	NULL	——	NULL	——	7663
DC4	024	DC4	024	STO	054	STO	064	7664
NAK	225	NAK	225	NAK	001	NAK	001	7665
SYN	226	SYN	226	IL	075	IL	075	7666
ETB	027	ETB	027	EOB	136	EOB	136	7667
CAN	030	CAN	030	DEL	177	DEL	137	7670
EM	231	EM	231	NULL	——	NULL	——	7671
SUB	232	SUB	232	NULL	——	NULL	——	7672
ESC	033	ESC	033	PF	076	PF	076	7673
FS	234	FS	234	NULL	——	NULL	——	7674
GS	035	GS	035	NULL	——	NULL	——	7675
RS	036	RS	036	NULL	——	NULL	——	7676
US	237	US	237	NULL	——	NULL	——	7677
NULL	——	NULL	——	NULL	——	NULL	——	7400
@	300	≤	044	@	020	≤	004	7401
^	336	¯	042	NULL	——	¯	020	7402
NULL	——	NULL	——	CNL	001	CNL	001	7403
:	072	:	276	:	153	:	121	7404
NULL	——	NULL	——	NULL	——	NULL	——	7405
NULL	——	NULL	——	NULL	——	NULL	——	7406
'	140	NULL	——	NULL	——	NULL	——	7407

3AE5A

NOS STANDARD CHARACTER SET

CDC GRAPHIC	ASCII GRAPHIC SUBSET	DISPLAY CODE	HOLLERITH PUNCH (026)	EXTERNAL BCD CODE	ASCII PUNCH (029)	ASCII CODE
:†	:	00†	8-2	00	8-2	3A
A	A	01	12-1	61	12-1	41
B	B	02	12-2	62	12-2	42
C	C	03	12-3	63	12-3	43
D	D	04	12-4	64	12-4	44
E	E	05	12-5	65	12-5	45
F	F	06	12-6	66	12-6	46
G	G	07	12-7	67	12-7	47
H	H	10	12-8	70	12-8	48
I	I	11	12-9	71	12-9	49
J	J	12	11-1	41	11-1	4A
K	K	13	11-2	42	11-2	4B
L	L	14	11-3	43	11-3	4C
M	M	15	11-4	44	11-4	4D
N	N	16	11-5	45	11-5	4E
O	O	17	11-6	46	11-6	4F
P	P	20	11-7	47	11-7	50
Q	Q	21	11-8	50	11-8	51
R	R	22	11-9	51	11-9	52
S	S	23	0-2	22	0-2	53
T	T	24	0-3	23	0-3	54
U	U	25	0-4	24	0-4	55
V	V	26	0-5	25	0-5	56
W	W	27	0-6	26	0-6	57
X	X	30	0-7	27	0-7	58
Y	Y	31	0-8	30	0-8	59
Z	Z	32	0-9	31	0-9	5A
0	0	33	0	12	0	30
1	1	34	1	01	1	31
2	2	35	2	02	2	32
3	3	36	3	03	3	33
4	4	37	4	04	4	34
5	5	40	5	05	5	35

3AE13A

† TWELVE OR MORE ZERO BITS AT THE END OF A 60-BIT WORD ARE INTERPRETED AS END-OF-LINE MARK RATHER THAN TWO COLONS.

CDC GRAPHIC	ASCII GRAPHIC SUBSET	DISPLAY CODE	HOLLERITH PUNCH (026)	EXTERNAL BCD CODE	ASCII PUNCH (029)	ASCII CODE
6	6	41	6	06	6	36
7	7	42	7	07	7	37
8	8	43	8	10	8	38
9	9	44	9	11	9	39
+	+	45	12	60	12-8-6	2B
−	−	46	11	40	11	2D
*	*	47	11-8-4	54	11-8-4	2A
/	/	50	0-1	21	0-1	2F
(	(	51	0-8-4	34	12-8-5	28
)	)	52	12-8-4	74	11-8-5	29
$	$	53	11-8-3	53	11-8-3	24
=	=	54	8-3	13	8-6	3D
BLANK	BLANK	55	NO PUNCH	20	NO PUNCH	20
,(COMMA)	,(COMMA)	56	0-8-3	33	0-8-3	2C
.(PERIOD)	.(PERIOD)	57	12-8-3	73	12-8-3	2E
≡	#	60	0-8-6	36	8-3	23
[	[	61	8-7	17	12-8-2	5B
]	]	62	0-8-2	32	11-8-2	5D
%†	%	63	8-6	16	0-8-4	25
≠	"(QUOTE)	64	8-4	14	8-7	22
→	_(UNDERLINE)	65	0-8-5	35	0-8-5	5F
∨	!	66	11-0	52	12-8-7	21
∧	&	67	0-8-7	37	12	26
↑	'(APOSTROPHE)	70	11-8-5	55	8-5	27
↓	?	71	11-8-6	56	0-8-7	3F
<	<	72	12-0	72	12-8-4	3C
>	>	73	11-8-7	57	0-8-6	3E
≤	@	74	8-5	15	8-4	40
≥	\	75	12-8-5	75	0-8-2	5C
¬	⌒(CIRCUMFLEX)	76	12-8-6	76	11-8-7	5E
;(SEMICOLON)	;(SEMICOLON)	77	12-8-7	77	11-8-6	3B

3AE6A

† IN INSTALLATIONS USING THE CDC 63-GRAPHIC SET, DISPLAY CODE 00 HAS NO ASSOCIATED GRAPHIC OR HOLLERITH CODE; DISPLAY CODE 63 IS THE COLON(8-2 PUNCH). THE SELECTION OF THE 63- OR 64-CHARACTER SET FOR TAPES IS AN INSTALLATION OPTION.

ASCII/DISPLAY CODE AND EBCDIC/DISPLAY CODE CONVERSION

DISPLAY CODE		ASCII				EBCDIC			
		UPPERCASE		LOWERCASE		UPPERCASE		LOWERCASE	
OCTAL	CHAR	CHAR	HEX	CHAR	HEX	CHAR	HEX	CHAR	HEX
00	:	:	3A	SUB	1A	:	7A	SUB	3F
01	A	A	41	a	61	A	C1	a	81
02	B	B	42	b	62	B	C2	b	82
03	C	C	43	c	63	C	C3	c	83
04	D	D	44	d	64	D	C4	d	84
05	E	E	45	e	65	E	C5	e	85
06	F	F	46	f	66	F	C6	f	86
07	G	G	47	g	67	G	C7	g	87
10	H	H	48	h	68	H	C8	h	88
11	I	I	49	i	69	I	C9	i	89
12	J	J	4A	j	6A	J	D1	j	91
13	K	K	4B	k	6B	K	D2	k	92
14	L	L	4C	l	6C	L	D3	l	93
15	M	M	4D	m	6D	M	D4	m	94
16	N	N	4E	n	6E	N	D5	n	95
17	O	O	4F	o	6F	O	D6	o	96
20	P	P	50	p	70	P	D7	p	97
21	Q	Q	51	q	71	Q	D8	q	98
22	R	R	52	r	72	R	D9	r	99
23	S	S	53	s	73	S	E2	s	A2
24	T	T	54	t	74	T	E3	t	A3
25	U	U	55	u	75	U	E4	u	A4
26	V	V	56	v	76	V	E5	v	A5
27	W	W	57	w	77	W	E6	w	A6
30	X	X	58	x	78	X	E7	x	A7
31	Y	Y	59	y	79	Y	E8	y	A8
32	Z	Z	5A	z	7A	Z	E9	z	A9
33	0	0	30	DLE	10	0	F0	DLE	10
34	1	1	31	DC1	11	1	F1	DC1	11
35	2	2	32	DC2	12	2	F2	DC2	12
36	3	3	33	DC3	13	3	F3	TM	13
37	4	4	34	DC4	14	4	F4	DC4	3C

3AE7A

DISPLAY CODE		ASCII				EBCDIC			
		UPPERCASE		LOWERCASE		UPPERCASE		LOWERCASE	
OCTAL	CHAR	CHAR	HEX	CHAR	HEX	CHAR	HEX	CHAR	HEX
40	5	5	35	NAK	15	5	F5	NAK	3D
41	6	6	36	SYN	16	6	F6	SYN	32
42	7	7	37	ETB	17	7	F7	ETB	26
43	8	8	38	CAN	18	8	F8	CAN	18
44	9	9	39	EM	19	9	F9	EM	19
45	+	+	2B	VT	0B	+	4E	VT	0B
46	−	−	2D	CR	0D	−	60	CR	0D
47	*	*	2A	LF	0A	*	5C	LF	25
50	/	/	2F	SI	0F	/	61	SI	0F
51	(	(	28	BS	08	(	4D	BS	16
52	)	)	29	HT	09	)	5D	HT	05
53	$	$	24	EOT	04	$	5B	EOT	37
54	=	=	3D	GS	1D	=	7E	IGS	1D
55	SP	SP	20	NUL	00	SP	40	NUL	00
56	,	,	2C	FF	0C	,	6B	FF	0C
57	.	.	2E	SO	0E	.	4B	SO	0E
60	≡	#	23	ETX	03	#	7B	ETX	03
61	[	[	5B	FS	1C	¢	4A	IFS	1C
62	]	]	5D	SOH	01	!	5A	SOH	01
63	%	%	25	ENQ	05	%	6C	ENQ	2D
64	≠	"	22	STX	02	"	7F	STX	02
65	→	_	5F	DEL	7F	_	6D	DEL	07
66	∨	!	21	}	7D	|	4F	}	D0
67	∧	&	26	ACK	06	&	50	ACK	2E
70	↑	'	27	BEL	07	'	7D	BEL	2F
71	↓	?	3F	US	1F	?	6F	IUS	1F
72	<	<	3C	{	7B	<	4C	{	C0
73	>	>	3E	RS	1E	>	6E	IRS	1E
74	≤	@	40	`	60	@	7C	`	79
75	≥	\	5C	¦	7C	\	E0	¦	6A
76	¬	^	5E	~	7E	¬	5F	~	A1
77	;	;	3B	ESC	1B	;	5E	ESC	27

3AE8A

CARRIAGE CONTROL CHARACTERS

CHARACTER	COMMAND
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
8	SKIP TO FORMAT CHANNEL 1 BEFORE PRINT †
7	SKIP TO FORMAT CHANNEL 2 BEFORE PRINT †
6	SKIP TO FORMAT CHANNEL 3 BEFORE PRINT †
5	SKIP TO FORMAT CHANNEL 4 BEFORE PRINT †
4	SKIP TO FORMAT CHANNEL 5 BEFORE PRINT †
3	SKIP TO FORMAT CHANNEL 6 BEFORE PRINT †
H	SKIP TO FORMAT CHANNEL 1 AFTER PRINT
G	SKIP TO FORMAT CHANNEL 2 AFTER PRINT
F	SKIP TO FORMAT CHANNEL 3 AFTER PRINT
E	SKIP TO FORMAT CHANNEL 4 AFTER PRINT
D	SKIP TO FORMAT CHANNEL 5 AFTER PRINT
C	SKIP TO FORMAT CHANNEL 6 AFTER 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

3AE9A

†No space after print. For all other control characters, a line feed is issued after print.

††Used only on the 512 and 580 line printers. The deselection of auto eject mode on a 512 or 580 line printer results in the deselection of 8 lines per inch, if previously selected.

DAYFILE MESSAGES B

This appendix contains an alphabetical listing of the messages which may appear in a user's dayfile. Lowercase characters are used to identify variable names or fields. If the first word or characters are variable, the message is listed according to the second word. For example, the message

pfn ALREADY PERMANENT, AT nnn.

is listed alphabetically with the messages beginning with the letter A. This is done because the variable pfn (permanent file name) may change each time the message is issued. All messages beginning with numbers follow the alphabetical listing.

The CIO and PFM file processors utilize the file environment table (FET) as a communication area to contain information about the requests of a user's job. Higher level languages (COBOL, FORTRAN, etc.) automatically establish and use these areas but the COMPASS programmer must define the FET. (Refer to volume 2 for detailed information on the FET.) CIO and PFM error messages contain the address, nnn, of the FET associated with the request and the logical file name, fff, from word zero of the table (FET+0).

When the error processing (ep) bit is set in word 1 of the FET, status information is returned by the function processor when an abnormal situation or error occurs. The abnormal termination codes are returned to bits 10 through 13 of word zero of the FET (bits 10 through 17 of PFM). Following the alphabetical listing of messages is a list of LFM and PFM error codes and explanations. Also included at the end of this section is a table summary of the action taken by PFM when an error is detected while reading mass storage.

Message	Routine	Description
ACCOUNT BLOCK LIMIT.	3AB	The monitor detected the expiration of the account block SRU limit.
ADDRESS ERROR.	TCS	CM address in call is beyond the field length.
ADDRESS OUT OF RANGE aaaaaa.	CPMEM	The address aaaaaa on a correction statement is greater than or equal to the user's field length. The correction statement is ignored and LOC continues.
pfn ALREADY PERMANENT, AT nnn.	PFM	The user has already saved or defined a file with the name specified.
ARG. ERROR.	LDR	LDR parameters were outside the user's field length.
ARGUMENT ERROR.	RESEX/ISF	A control statement is syntactically incorrect. Recheck parameters. On tape management statements, the system issues this message if both ring enforcement options (PO=R and PO=W) or more than one EOT option (PO=I, PO=P, PO=S) is specified.
ARITH, ERROR x AT yyyyyy.	3AB	The monitor detected an arithmetic error condition x at address yyyyyy.
BAD DECK NAME.	TCS	A deck name of more than 7 characters was encountered.
BINARY SEQ. ERROR, RECxxxx CDyyyy.		A binary card was found to be out of sequence and the job was terminated. xxxx Number (in octal) of record in which sequence error occurred. yyyy Number (in octal) of card within the record which caused the sequence error.
BLANK TAPE, fff AT nnn.	1MT	A blank tape was encountered on a read operation. (Blank tape is defined as more than 25 feet of erased tape.)
BLOCK COUNT ERROR IN TRAILER LABEL, fff AT nnn.	1MT	The block count in the EOF1 or EOV1 label did not match the block count maintained by the tape executive during the read operation.
BLOCK LENGTH ERROR ON fff AT nnn.	1MT	The software-recorded block length did not match the length of the block read (this message applies to I format tapes only).
BLOCK SEQUENCE ERROR, fff AT nnn.	1MT	The software-recorded block length did not match the length of the block read, or the block number did not match the software-record block number (this message applies to I format tapes only).
BLOCK TOO LARGE ON fff AT nnn.	1MT	The tape being read contained a data block greater in size than that allowed by the specified format or by user declaration (this message applies to S or L format tapes only).
BOT/EOT ENCOUNTERED, fff AT nnn.	1MT	Indicates an abnormal tape position.
BUFFER ARG. ERROR.	TCS	CM address in call is not less than the field length minus the word count; buffer extends past the job's field length.
BUFFER ARGUMENT ERROR.	QFM	A buffer pointer did not conform to the following constraints. FIRST ≤ IN FIRST ≤ OUT OUT < LIMIT ≤ FL

Message	Routine	Description
BUFFER ARGUMENT ERROR ON fff AT nnn.	CIO/1MT	A buffer pointer did not conform to the following constraints. FIRST ≤ IN FIRST ≤ OUT OUT < LIMIT ≤ FL The system provides a dump of the FET on file OUTPUT.
BUFFER CONTROL WORD ERROR ON fff AT nnn.	CIO/1MT	The block length specified during a write operation was greater than the allowable PRU size for the device. For tape operations, this message can also indicate that the unused bit count is illegal or that an attempt was made to write a record shorter than the noise record size.
pfn BUSY, AT nnn.	PFM	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
n CARD(S) NOT PROCESSED.		Errors on n directives prevented them from being processed.
CATALOG OVERFLOW - FILES, AT nnn.	PFM	The number of files in the user's catalog exceeds his limit (refer to LIMITS control statement, section 6).
CATALOG OVERFLOW - SIZE AT nnn.	PFM	The cumulative size of the indirect access files in the user's catalog exceeds his limit (refer to LIMITS control statement, section 6).
CHANNEL MALFUNCTION, fff AT nnn.	1MT	Hardware malfunction.
CHARGE ABORTED.	CHARGE	A central site operator action caused the CHARGE operation to abnormally terminate. Resubmit job.
CHARGE FILE BUSY.	CHARGE	The file which the system uses to validate charge number and project number is busy. Resubmit job.
CHARGE ILLEGAL AT THIS HOUR.	CHARGE	The specified project number cannot be used at this time of the day.
CHECKPOINT nnnn COMPLETE.	CHKPT	Indicates that checkpoint nnnn has been 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 were specified.
CHECKPOINT nnnn COMPLETED TO xxxxxxx.	CHKPT	Indicates that checkpoint nnnn has been completed to file xxxxxxx. Issued if alternate CB checkpoint files are used.
CHECKPOINT FILE ERROR.	CHKPT/RESTART	During a restart operation, either the checkpoint file lfn specified on the RESTART control statement was empty or RESTART detected a format error attempting to read the specified checkpoint file.
CHECKPOINT NOT FOUND.	RESTART	The specified checkpoint (nn parameter on RESTART statement) could not be found on the file.
CKP REQUEST.	CHKPT	A checkpoint has been initiated.
CM NOT VALIDATED.	TCS	The number of CM words specified on the job statement exceeds that for which the user is validated.

Message	Routine	Description
COMPILER NOT IN LIBRARY.	TCS	An LDC control statement requested loading of a compiler not on the system.
CONTROL CARD ARGUMENT ERROR.	QFSP	An invalid argument was encountered on a control statement.
CONTROL CARD ERROR.		Loader failed to find the requested file.
CONTROL STATEMENT LIMIT.	1AJ	The number of control statements processed for a job has exceeded the limit for which the user is validated.
CONVERSION NOT FOUND.	PFM	The conversion table specified by the TS option was not found.
CONVERSION NOT SPECIFIED.		Neither a TS nor 64 option was specified on a CONVERT control statement.
COPY COMPLETE.	COPY	Informative message issued when a system file copy is complete.
CORE OVERFLOW, JOB ABORTED.	CPM	Table overflow occurred; rerun using more central memory field length.
CPxx,.... .		Refer to the EQxx,... series of corresponding messages for full descriptions of messages beginning with CPxx,.... .
CPM ARG. ERROR.	CPM	Error(s) encountered and job aborted.
CPM ILLEGAL REQUEST.	CPM	A CPM function was issued without the auto recall specified or job was not of system origin.
CPU ERROR EXIT xx AT yyyyyy.	1AJ	Monitor has detected a CPU error exit condition xx at address yyyyyy (refer to Error Control, section 3).
CRxx,.... .		Refer to the EQxx,... series of corresponding messages for full descriptions of messages beginning with CRxx,.... .
CUX - ILLEGAL USER ACCESS		Account not validated for system privileges.
DATA BASE ERROR.	CHARGE/ MODVAL	One of the following: The system detected an error in its validation file. The user should contact installation personnel.
DATA BASE ERROR n.	PROFILE	An abnormal error has been detected. Notify the analyst.
DATA TRANSFER ERROR, AT nnn.	PFM	An error occurred in a read operation during a file transfer.
DAYFILE TERMINATED	SFM	Informative message issued to the terminated dayfile.
xxxxxx DAYFILE TERMINATED.	SFM	Informative message indicating dayfile xxxxxx has been terminated (issued to system and control point dayfiles).

Message	Routine	Description
DExx, Cyy, ec, ann, Stttt, Axxxxxxx.	6DE	An error has been detected on extended core storage. The nature of the error is determined by examining each parameter in the message. xx EST ordinal of ECS unit yy Channel number ec Error code; one of the following: PE Parity error/checkword error AD Address error ST Device status error RS Device reserved NR Device not ready a Type of operation; one of the following: R Read W Write nn Retry count; error is considered irrecoverable after the following number of retries: PE 10 AD 10 ST 64 FN 10 RS Indefinite NR Indefinite tttt Device status; implies there was an incomplete transfer if status does not indicate an error Axxxxxxx Physical address at beginning of block
DEMAND EXCEEDED.	RESEX	The user attempted to assign more units than he scheduled on the RESOURC statement.
DEMAND INSTALLATION ERROR.	RESEX	The user requested more units than exist at the installation.
DEMAND VALIDATION ERROR.	RESEX	The specified number of units exceeds the user's validation limits.
DENSITY CHANGE, fff AT nnn.	1MT	The density of the tape changed during a read or write operation. If this error occurs on the first block on the tape, the additional message DENSITY SPECIFIED DIFFERENT FROM TAPE. is issued. In requesting the tape, the user should specify the density in which the tape was written.
DEVICE ERROR ON FILE fff AT nnn.	CIO	An irrecoverable error occurred on the mass storage device containing the file fff.
DEVICE UNAVAILABLE, AT nnn.	PFM	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.

Message	Routine	Description
DIxx, Cyy, ec, ann, Stttt, FNqqqq. or DIxx, Cyy, ec, ann, Stttt, Uuu Ccccc Sttss.	6DI	An error has been detected on mass storage device xx. The nature of the error is determined by examining each parameter in the message. xx EST ordinal of 844-21 disk yy Channel number ec Error code (one of the following): PE Parity error/checkword error AD Address error ST Device status error FT Function timed out with no response RS Device reserved NR Device not ready CR Controller reserved a Type of operation (one of the following): R Read W Write nn Retry count; error is considered irrecoverable after the following number of retries: PE 10 AD 10 ST 64 FT 3 RS Indefinite NR Indefinite CR Indefinite tttt Device status; implies there was an incomplete transfer if status does not indicate an error qqqq Function which timed out Uuu Physical unit Ccccc Physical cylinder Sttss tt Physical track ss Physical sector (Uuu, Ccccc, Sttss: Physical address)
DIRECT ACCESS DEVICE ERROR, AT nnn.	PFM	The specified file 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: • The device is not specified as a direct access device in the catalog descriptor 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 ERRORS.	PROFILE/ MODVAL/ OPLEDIT	An invalid directive statement was encountered. If the message is issued in response to a PROFILE request, refer to the specific directive errors as listed for output file diagnostics.
n DIRECTIVE ERROR(S).	PROFILE	Occurs when there are conflicts or omissions in implied deletes or insertions.
DIRECTIVE CARD ERROR.	PROFILE	Occurs when an illegal directive statement is encountered; for example, syntax error.

Message	Routine	Description
DJxx, Cyy, ec, ann, Stttt, FNqqqq. or DJxx, Cyy, ec, ann, Stttt, Uuu Ccccc Sttss.	6DJ	An error was detected on mass storage device xx. The nature of the error is determined from the parameters in the message. xx EST ordinal of 844-41 disk yy Channel number ec Error code (one of the following): PE Parity error/checkword error AD Address error ST Device status error FT Function timed out with no response RS Device reserved NR Device not ready CR Controller reserved a Type of operation (one of the following): R Read W Write nn Retry count; error is considered irrecoverable after the following number of retries: PE 10 AD 10 ST 64 FT 3 RS Indefinite NR Indefinite CR Indefinite tttt Device status; implies there was an incomplete transfer if the status does not indicate an error qqqq Function which timed out Uuu Physical unit Ccccc Physical cylinder Sttss tt Physical track ss Physical sector } Physical address

Message	Routine	Description
DPxx, Cyy, ec, ann, Stttt, Axxxxxxx.	6DP	An error has been detected on distributive data path (DDP). The nature of the error is determined by examining each parameter in the message. xx EST ordinal of DDP/ECS yy Channel number ec Error code (one of the following): PE Parity error/checkword error AD Address error ST Device status error RS Device reserved NR Device not ready a Type of operation (one of the following): R Read W Write nn Retry count; error is considered irrecoverable after the following number of retries: PE 10 AD 10 ST 64 FN 10 RS Indefinite NR Indefinite tttt Device status; implies there was an incomplete transfer if status does not indicate an error Axxxxxxx Physical address at beginning of block
DSP - CAN NOT ROUTE JOB INPUT FILE.	DSP	The job input file cannot be routed.
DSP - COMPLETE BIT ALREADY SET.	DSP	The complete bit was not cleared before DSP was called.
DSP - DEVICE UNAVAILABLE	DSP	DSP attempted to create a file on a device that was turned off or currently unavailable for access.
DSP - FILE NAME ERROR.	DSP	An attempt was made to create a file with an invalid file name.
DSP - FILE NOT ON MASS STORAGE.	DSP	An attempt was made to route a file not on mass storage.
DSP - FILE ON REMOVABLE DEVICE.	DSP	A file on a removable device cannot be routed.
DSP - FORMS CODE NOT ALPHANUMERIC.	DSP	Forms code must consist of two alphanumeric characters.
DSP - FNT/DEVICE FULL.	DSP	There is no space in the FNT or on the device for current use.
DSP - ILLEGAL FILE TYPE.	DSP	The file being processed is not a PRFT, PHFT, INFT, or LOFT file type.
DSP - ILLEGAL ORIGIN TYPE	DSP	DSP cannot route the file to the input queue with the origin type specified by the caller.
DSP - ILLEGAL REQUEST.	DSP	One of the following: 1. DSP was not called with recall (does not apply when queue priority is greater than MXPS). 2. Parameter list address was out of range.

Message	Routine	Description
DSP - ILLEGAL USER CARD.	DSP	User attempted to route a file with an illegal USER statement to the input queue.
DSP - IMMEDIATE ROUTINE - NO FILE.	DSP	The specified file for the immediate routing could not be found.
DSP - INVALID DISPOSITION CODE.	DSP	Specified disposition code is not recognized.
DSP - INVALID EXTERNAL CHARACTERISTICS	DSP	Caller specified an undefined external characteristic code.
DSP - INVALID TID.	DSP	One of the following: 1. User number and family name parameters were not in CM field length. 2. TID is greater than or equal to IDLM for batch jobs. 3. User number specified in parameter block does not compare with user number in control point area.
DSP - I/O SEQUENCE ERROR.	DSP	A request was made on a busy file.
DSP - LOCAL FILE LIMIT.	DSP	User has exceeded his/her local file validation limits.
DSP - OUTPUT FILE LIMIT.	DSP	Caller has exceeded his/her output file validation.
DSP - ROUTE TO INPUT NOT IMMEDIATE.	DSP	Routing a file to the input queue must be immediate.
DSP - THIS ROUTING NOT ALLOWED.	DSP	An attempt was made to change the origin type or queue type of a deferred routed file.
DSP - TOO MANY DEFERRED BATCH JOBS.	DSP	User has more jobs in the system than allowed. This check is ignored for users with system origin privileges.
DUMP FWA .GE. LWA+1.	CPMEM	The first word address of memory to be dumped was greater than the last word address plus 1 of memory.
DUPLICATE COMMON FILE NAME	LFM	A file of the same name as that specified in a COMMON or STAGE request already exists.
DUPLICATE FILE NAME.	LFM	The file specified already exists in the system.
DUPLICATE LINES.		Lines being dumped during a DMP operation were duplicated and suppressed.
DUPLICATE PROJECT NUMBER.	PROFILE	During a create run, PROFILE detected two or more identical project numbers within one charge number entry. The first project number is retained; all subsequent duplicate numbers are disregarded. All other project numbers are processed normally.
DUPLICATE USER NUMBER.	PROFILE	This message is printed if PROFILE detected two or more identical user numbers in one project number entry, or the user attempts to update the project profile file by adding a user number that already exists under the specified project number. The entire proejct number entry containing the duplicate user numbers is disregarded.

Message	Routine	Description
ECS LOAD ERROR.	3AE	Bad load address from ECS.
EDITING COMPLETE.		Informative message.
lfn EMPTY, AT nnn.	PFM	The file specified on a SAVE request contains no data.
EMPTY CATALOG.	CATLIST	No entries are present in the catalog.
EMPTY SORT INPUT FILE.	MSORT	File lfn specified on the SORT control statement contains no data.
END OF INFORMATION ENCOUNTERED.	COPY	Informative message issued when a local file copy is completed.
END OF TAPE, fff AT nnn.	1MT	The end of tape was encountered.
ENQUIRY COMPLETE.	ENQUIRE	Informative message issued when processing of ENQUIRE control statement is completed.
ENTRY POINT NOT FOUND.	3AD	The specified entry point could not be found on the overlay file.
EOF ENCOUNTERED BEFORE TERMINATION.		An end-of-file was encountered on a CONVERT input file before the specified record count was reached.
EOI ENCOUNTERED BEFORE TERMINATION.		An end-of-information was encountered on a CONVERT input file before the specified record count was reached.
EQ, Ccc-e-uu, vsn, rw, est, Sss, $scon_1$, $scon_2$. EQ, Ccc-Fff, Iii, Bnnnnnn, Lbbbb, Ppppppppp. EQ, Ccc, Eec, H000000000, type.	1MT	Three-line message describing a magnetic tape hardware malfunction occurring on a 657 or 659 tape unit. EQ MT for 657; NT for 659 The first line 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 the 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. est EST ordinal of the unit on which the tape was written. This is provided only for labeled tapes generated under NOS 1.0; otherwise, the field is blank. ss Status of the 6681/6684 interface. First digit represents a^{00} where bit $a=2^{11}$ of status; second digit represents bits 2^2-2^0 of status. $scon_1$ Status of the tape controller. $scon_2$ Status-2 of the controller, if available. The second line of the message contains: 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. 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 1MT internal error parameters.

Message	Routine	Description
		The third line of the message contains the following information. cc Channel number; the channel number is repeated to allow the analyst to associate this message with the first and second messages if errors are occurring on more than one tape channel at the same time. ec Octal error code value. 0000000000 Controller options selected at the time of the error; each two digits is a function code. type Additional description of the error (one of the following): BAD ERASE. Error detected after an erase was attempted to recover a write error. BLOCK TOO LARGE Data block was larger than expected. BUSY. Unit was still busy after 1 second. CHANNEL ILL. Channel is not accepting function or status requests properly. CON. REJ. Connect reject; unable to connect to the unit. CON. REJ. OFF. Connect reject; unable to connect to unit. Unit turned OFF. DENSITY CHANGE. Either user error where auto select does not match user selection (0-track only) or hardware error where status does not match user selection. FNnn, Pyyyy. Function nn was rejected by the controller; yyyy is the address in 1MT where the function was initiated. 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. NO EOP. No end-of-operation detected from unit within 1 second. NOISE. A noise block was skipped on the tape. NOT READY. Tape unit dropped ready status. ON THE FLY. Error was corrected as the data was read. POSITION LOST. The last good block written cannot be found during write recovery. RECOVERED. Previously reported error has been successfully recovered. STATUS. Error type cannot be determined so actual controller status is returned. WRONG PARITY. Tape was written in parity opposite that being read.
EQ, Ccc-uu, vsn, rw, est, Ss, GSgggg. EQ, Ccc, Dddd... d EQ, Ccc, Fff, Iii, Bnnnnnn, Lbbbb, Pppppppp. EQ, Ccc, Eec, Hhhhhhhhh, type.	1MT	Four-line message describing a magnetic tape hardware malfunction occurring on a 667 or 669 tape unit. EQ MT for 667; NT for 669 The first line provides the following information. cc-uu Channel and physical unit number of tape unit on which error was encountered. vsn Volume serial number associated with the tape on the specified unit.

Message	Routine	Description
		rw — Read (RD) or write (WR) operation; any operation not involving an actual read or write is listed as a read. est — EST ordinal of the unit on which the tape was written. This is provided only for labeled tapes generated under NOS 1.0; otherwise, the field is blank. s — Channel status. gggg — General status of magnetic tape unit. The second line of the message contains: 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 contains: 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 — 1MT internal error parameters. The fourth line of the message contains: cc — Channel number; repeated to associate this message with the previous messages. ec — Octal error code value. hhhhhhhh — Unit format parameters. Refer to Magnetic Tape Subsystem Reference Manual for descriptions of unit format parameter fields. type — Additional description of the error (one of the following): BAD ERASE. — Error detected after an erase was attempted to recover a write error. B.C. RESTART. — Magnetic tape controller firmware restarted. BLOCK TOO LARGE. — Data block was larger than expected. BUSY. — Unit was still busy after 1 second. CHANNEL ILL. — Channel is not accepting function or status requests properly. CON. REJ. — Connect reject; unable to connect to the unit. CON. REJ. OFF. — Connect reject; unable to connect to unit. Unit turned OFF. DENSITY CHANGE. — Either user error where auto select does not match user selection (9-track only) or a hardware error where status does not match user selection. FNnn, Pyyyy. — Function nn was rejected by the controller; yyyy is the address in 1MT where the function was initiated. 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.

Message	Routine	Description
		NO EOP. No end-of-operation detected from unit within 1 second. NOISE. A noise block was skipped on the tape. NOT READY. Tape unit dropped ready status. ON THE FLY. Error was corrected as the data was read. POSITION LOST. The last good block written cannot be found during write recovery. RECOVERED. Previously reported error has been successfully recovered. STATUS. Error type cannot be determined so actual controller status is returned. WRONG PARITY. Tape was written in parity opposite that being read.
EQxx, CHyy Adddd INCOMPLETE TRANSFER.	1IO	An incomplete data transfer was detected by a local batch equipment driver. EQ One of the following equipment types: CP 415 card punch CR 405 card reader LP 512 or 580 line printer LQ 512 line printer LR 580 line printer xx EST ordinal of local batch equipment yy Channel number dddd Octal byte count not transferred
EQxx, CHyy CONTROLLER HUNG BUSY.	1IO	The specified local batch controller did not drop BUSY status. EQ One of the following equipment types: CP 415 card punch CR 405 card reader LP 512 or 580 line printer LQ 512 line printer LR 580 line printer xx EST ordinal of local batch equipment yy Channel number
EQxx, CHyy Fzzzz FUNCTION TIMEOUT.	1IO	No response (inactive) was received after issuing a function code 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 512 or 580 line printer LQ 512 line printer LR 580 line printer xx EST ordinal of local batch equipment yy Channel number zzzz Function code

Message	Routine	Description
EQxx, CHyy Fzzzz REJ Paaaa, Cbbbb, Ecccc.	1IO	Detected function reject or transmission parity error on the specified local batch equipment. EQ One of the following equipment types: CP 415 card punch CR 405 card reader LP 512 or 580 line printer LQ 512 line printer LR 580 line printer xx EST ordinal of local batch equipment yy Channel number zzzz Function code aaaa Driver (1CD) address bbbb Converter status cccc Equipment status
EQxx, CHyy RESERVED.	1IO	The specified local batch equipment is reserved and cannot be connected on channel yy. EQ One of the following equipment types: CP 415 card punch CR 405 card reader LP 512 or 580 line printer LQ 512 line printer LR 580 line printer xx EST ordinal of local batch equipment yy Channel number
EQxx, CHyy TURNED OFF.	1IO	The specified local batch equipment was logically turned off (OFF status set in EST). Note that this message is preceded in the error log by a message for the same equipment which specifies the failing condition. EQ One of the following equipment types: CP 415 card punch CR 405 card reader LP 512 or 580 line printer LQ 512 line printer LR 580 line printer xx EST ordinal of local batch equipment yy Channel number
EQxx, DNdn, DIRECT ACCESS FILE ERROR, AT nnn.	PFM	The system sector data for the file does not match the catalog data. xx EST ordinal of device dn Device number
EQxx, DNdn, FILE LENGTH ERROR, AT nnn.	PFM	The length of a file does not equal the catalog length. xx EST ordinal of device dn Device number

Message	Routine	Description
		The action taken depends on the type of command issued. Command — Action 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 nnn.	PFM	An error was encountered in reading a portion of the permanent file catalog or permit information. xx EST ordinal of device dn Device number
EQxx, DNdn, RANDOM INDEX ERROR, AT nnn.	PFM	The random disk address of the permit sector is in error. xx EST ordinal of device dn Device number
EQxx, DNdn, REPLACE ERROR, AT nnn.	PFM	The same file was found twice during a catalog search. This error can occur for APPEND or REPLACE commands after a file is found and purged and the catalog search is continued. xx EST ordinal of device dn Device number
EQxx, DNdn, TRACK LIMIT, AT nnn.	PFM	No allocatable tracks remain on equipment xx. xx EST ordinal of device dn Device number

Message	Routine	Description
EQxx, RM=mmmmmmm, PF=ppppppp, UI=iiiiii.	PFM	Additional line written only in error log after one of the following messages. EQxx, DNdn, DIRECT ACCESS FILE ERROR, AT nnn. EQxx, DNdn, FILE LENGTH ERROR, AT nnn. EQxx, DNdn, MASS STORAGE ERROR, AT nnn. EQxx, DNdn, RANDOM INDEX ERROR, AT nnn. EQxx, DNdn, REPLACE ERROR, AT nnn. EQxx, DNdn, TRACK LIMIT, AT nnn. xx EST ordinal of device mmmmmmm Family name ppppppp Permanent file name iiiiii User index
EQUIPMENT NOT AVAILABLE	LFM/RESEX	Requested equipment is either in use or does not exist.
ERASE LIMIT, fff AT nnn.	1MT	The system made 20 erasures (10 feet of tape) without being able to successfully write the tape.
ERROR AT LINE xxx		Issued when errors occur while resequencing a BASIC program. The line containing the error is specified by xxx.
ERROR CODE xx, lfn AT addr.	1MT	1MT error code xx has occurred but no specific message is issued. This would normally not occur unless the job was dropped by the operator.
ERROR IN ARGUMENTS		One or more of the following conditions 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 packname or auxiliary device parameter were selected; auxiliary devices do not have device numbers.
ERROR IN COMMAND PARAMETERS.		Either no parameters are allowed or an illegal parameter has been encountered.
ERROR IN DATE.	PURGALL	The format of the date (ad, md, or cd) parameter in a PURGALL request was incorrect.
ERROR IN DEVICE NUMBER.	PURGALL	The file residency as specified by the device number parameter was illegal.
ERROR IN DIRECTORY.		Program library does not have a directory record or has an incorrectly formatted directory record.
ERROR IN FILE ARGUMENTS	FILES	The parameter could not be recognized.

Message	Routine	Description
ERROR IN FILE CATEGORY.	PURGALL/ PFILES	The user specified an illegal file category.
ERROR IN FILE TYPE.	PURGALL	The user specified an illegal file type.
ERROR IN IDENTIFIER.	PROFILE/ MODVAL	PROFILE cannot recognize a directive identifier. The action taken depends upon the position of the erroneous identifier within the entry. ● If the error occurs within a project number entry, the entire project number entry is disregarded. ● If the error occurs in a directive that appears after a charge number but before the first project number, only the erroneous directive is disregarded. However, if the error occurs on the first PN directive, the entire project number entry is disregarded. ● If the error occurs in any PN directive except the first one, it is treated as an error within the preceding project number entry. Both the project number entry for the erroneous project number and the preceding project number entry are disregarded.
ERROR IN ROUTE FUNCTION, LFN =filenam.	DSP	Informative message issued to the system dayfile stating an error occurred while routing filenam.
ERROR IN LIMITS ARGUMENT.		Parameters were included on the LIMITS statement.
ERROR IN NUMERIC DATA.	PROFILE/ MODVAL	PROFILE detected nonnumeric data or numeric data exceeding the maximum limit for specified control value. The entire project number entry containing the erroneous directive is disregarded.
ERROR IN PASSWOR ARGUMENTS.		Parameters specified on a PASSWOR control statement were in error.
ERROR IN PROFILE ARGUMENTS	PROFILE	Error on PROFILE control statement.
ERROR IN TIME.	PURGALL	The format of the time parameter in a PURGALL request was incorrect.
ERROR - FILE(S) NOT PROCESSED.	CHKPT	One or more files were not checkpointed because CHKPT detected address errors.
FAST-ATTACH PROFILE FILE ILLEGAL.	PROFILE	Project file cannot be in fast-attach status on a reformat run.
FET ADDRESS OUT OF RANGE AT nnn.	CIO	FET extends past job's field length.
FET PARAMETER ERROR ON fff AT nnn.	CIO	One of the parameters in the FET is illegal or the FET is not long enough for the parameter.
nnnn FILE DEQUEUED Dndn FMxxxxxxx.	QREC	Indicates the number of files that have been dequeued on the specified device. nnnn Number of files dn Device number xxxxxxx Family name
FILE EMPTY.	LFM/SFM/QFM	The file specified was empty.

Message	Routine	Description
FILE ERROR lfn.	CHKPT/ RESTART	An illegal address was detected on file lfn.
FILE NAME ERROR, AT nnn.	PFM	File name contains illegal characters.
FILE NOT FOUND	LFM/SFM/QFM	Requested file was not found.
FILE NOT ON MASS STORAGE.	3AD	The specified file does not reside on mass storage.
FILE NOT OVERLAY FORMAT.	LDR	The first record of the file was not an overlay.
FILE TOO LONG, AT nnn.	PFM	The local file specified for a SAVE, REPLACE, or APPEND command exceeds the length allowed, or the direct access file specified for an ATTACH in WRITE, MODIFY, or APPEND mode exceeds the direct access file length limit for which the user is validated.
FL BEYOND MFL.	1MA	Field length requirements for the job step exceed the field length allowed. The user will have to increase the job step field length.
FL TOO SHORT FOR LIBRARY GENERATION.	LIBGEN	Additional memory is required for LIBGEN.
FL TOO SHORT FOR PROGRAM.	3AE	The user's field length is too short for the program.
FM NOT LEGAL FAMILY.	PROFILE	Illegal family name is specified with FM parameter.
pfn FOUND, AT nnn.	PFM	The specified permanent file was found.
FORMAT ERROR ON CONTROL CARD.	TCS	An error was detected in the format of the control statement.
FORMAT ERROR ON OVERLAY DIRECTIVE	LDR	Illegal overlay directive parameter or no arguments found.
FNT IS FULL.	QFM	The FNT filled during processing of the requeue function and all files could not be requeued.
ILLEGAL ACCESS TO EXECUTE ONLY FILE.	3AD	The specified file is an execute-only file.

Message	Routine	Description
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.
ILLEGAL CHARGE.	CHARGE	The specified charge or project number does not exist or the project number was not assigned to this user.
ILLEGAL COMMON MEMORY MANAGER REQUEST.	1MA	Memory request with reserved bits in the parameter block were set incorrectly.
ILLEGAL CONTROL CARD.	TCS/RESEX	One of the following: • The control statement could not be identified. • An invalid parameter was specified or no terminator was detected. • The user attempted to pass too many parameters on the program call statement (such as LGO). • The user submitted a control statement considered illegal because of his validation [for example, if access option 1 (refer to LIMITS control statement) was not set and the user submitted a PASSWOR control statement)]. • The user submitted a control statement considered illegal for a particular job t̬pe or file type (for example, the use of a FAMILY statement in a nonsystem origin job).
ILLEGAL COUNT.	COPYBF/COPYBR/ COPYX	Number of files in copy request was either illegal or zero.
ILLEGAL DEVICE REQUEST, AT nnn.	PFM	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.
ILLEGAL DISPOSE CODE.		The queue type (q_i) specified on a DISPOSE control statement was unrecognizable.
ILLEGAL EQUIPMENT.		File is assigned to illegal equipment for the specific request (for example, the file specified in a COMMON request is not on mass storage).
ILLEGAL EXTENSION OF fff AT nnn.	CIO	The user attempted to lengthen a file that could not be extended.
ILLEGAL FILE NAME fff AT nnn.	CIO	The file name does not conform to established rules.
ILLEGAL FILE TYPE.	LFM/QFM	The specified file is of a type not allowed in the requested operation. For example, this message would be issued if the file name in a RELEASE request was not a queue type file (input, print, or punch) or if the user attempted to make a nonlocal file a library file.
pfn ILLEGAL FILE TYPE, AT nnn.	PFM	The user attempted to DEFINE a local file residing on a device other than a permanent file device.

Message	Routine	Description
ILLEGAL INPUT FILE.	PACK	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 represents data typed at the terminal keyboard, and therefore, cannot be packed.
ILLEGAL I/O REQUEST ON FILE fff AT nnn.	CIO	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 on the FET on file OUTPUT.
ILLEGAL LEVEL NUMBER.	LDR	One of the following: ● Assembly error ● Level number greater than 77_8 ● First overlay not zero level (0,0) overlay
ILLEGAL LOAD ADDRESS.	3AE	The load address is less than 2.
ILLEGAL MODIFICATION OF fff AT nnn.	CIO	Either the user has attempted to shorten a modify-only file or the file cannot be modified at all.
ILLEGAL ORIGIN.	SFM/QFM	The origin type specified when releasing a local file to a queue was illegal.
ILLEGAL ORIGIN SPECIFIED.		Origin word count error.
ILLEGAL PROFILE INQUIRE.	PROFILE	The user is not allowed to access the control information for the charge number supplied.
ILLEGAL RECORD TERMINATION.	COPYX	Illegal format on record terminator.
ILLEGAL SORT PARAMETER.		The SORT control statement is in error.
ILLEGAL TERMINAL REQUEST.		A command intended for time-sharing origin jobs only (refer to Time-Sharing Commands, section 4) has been used in a non-time-sharing origin job.
ILLEGAL USER ACCESS.	LFM/QFM/QFSP/RESEX	User tried to perform an operation for which he was not validated.
ILLEGAL USER ACCESS, AT nnn.	PFM	The user is not validated to create direct access or indirect access files or to access auxiliary devices.
ILLEGAL USER CARD	CPM	User number or password could not be validated, or a secondary user statement was encountered while secondary user statements were disabled.
IMPROPER ACCESSIBILITY.	RESEX	The user did not specify the correct file accessibility on the LABEL statement, or volume accessibility was set and a nonsystem origin user attempted to assign the tape as unlabeled.
IMPROPER VALIDATION	TCS	A validation program (one containing a VAL= entry point, such as that used for CHARGE and USER) is required before continuing.
INDEX ADDRESS OUT OF RANGE FOR fff AT nnn.	CIO	The random sector address for a random input/output request was equal to or greater than field length.
INPUT FILE IN NORERUN STATUS.	QFM	Informative message.
INPUT FILE IN RERUN STATUS.	QFM	Informative message.

Message	Routine	Description
INQUIRY COMPLETE.	MODVAL	The inquiry was successfully completed.
deckname-INVALID CS, 63 ASSUMED.	TCS	Character set identification for deck deckname was not recognizable. OPLEDIT assumes 63-character set and uses it for the new program library if one is being created.
INVALID USER ACCESS - CONTACT SITE OPR.	CPM, 1JA, 1LS	The user number specified has exhausted its security count. The user number will be denied all access to the operating system until the security count has been reset by the operator.
I/O ON EXECUTE-ONLY FILE fff AT.nnn.	CIO	The user attempted to read, write, or position an execute-only file. RETURN is the only operation allowed for an execute-only file.
I/O SEQUENCE ERROR.	QFM	Action was requested by a busy file.
I/O SEQUENCE ERROR, AT nnn.	PFM	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.
I/O SEQUENCE ERROR ON FILE fff AT nnn.	CIO	The user attempted to perform more than one concurrent function on a single file.
JOB ABORTED, fff AT nnn.	1MT	The job was aborted while a tape operation was pending.
JOB CARD ERROR. (20 characters)	3AA	The job statement on the file being submitted is in error. The first 20 characters of the statement in error follow the message.
JOB EXECUTING.		The job is either executing or has been rolled out for a higher priority job.
JOB IN INPUT QUEUE.		Informative message.
JOB IN NORERUN STATE ON RECOVERY.	1AJ	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).
JOB IN OUTPUT QUEUE.		Informative message.
JOB IN PUNCH QUEUE.		Informative message.
JOB NOT FOUND.		This message normally indicates that the job has been processed and no longer exists in the system. However, it may also be issued if the jobname was entered incorrectly (misspelled).
JOB STEP LIMIT.	3AB	The monitor detected the expiration of the job step SRU limit.
JOB REPRIEVED.	SFP	The job has been successfully reprieved.
LABEL CONTENT ERROR, fff AT nnn.	1MT	A block read was the correct size for a label but one or more required fields (such as the label name) were incorrect. The programmer should use the LISTLB control statement to determine the cause of the problem.
LABEL MISSING, fff AT nnn.	1MT	During a read operation, a required label was missing. The programmer should use the LISTLB control statement to determine the cause of the problem.
LABEL PARAMETER CONFLICT ON OPEN, fff AT nnn.	1MT	Label fields did not match on open request. An additional message FIELD BEGINNING AT nnn NO COMPARE. specifying the decimal character position in HDR1 of the first field that did not compare correctly is also issued.

Message	Routine	Description
LBC ARGUMENT ERROR.	CPMEM	The load address, addr, specified on the LBC control statement was nonnumeric.
LBC FWA .GE. FL.	CPMEM	The load address specified on the LBC control statement was greater than or equal to the user's field length.
LDR ERROR.	LDR	Issued after one of the following errors: OVERLAY NOT FOUND IN LIBRARY. ARG ERROR. FILE NOT OVERLAY FORMAT.
LEVEL NUMBER MISSING	LDR	
LFM ILLEGAL REQUEST.	LFM	One of the following: ● LFM function detected was not recognized as a legal function. ● An LFM function was issued without the auto recall bit set.
LIBGEN ARGUMENT ERROR.	LIBGEN	An invalid parameter was used on the LIBGEN control statement.
LIBRARY GENERATION COMPLETE.	LIBGEN	Informative message.
LIBRARY GENERATION FILE EMPTY.	LIBGEN	The file to be processed is empty.
LINE NUMBER LIMIT EXCEEDED.	RESEQ	The line number encountered or required during a resequencing (RESEQ) operation exceeded 99999.
LOADER MISSING.	TCS	Either CALL or LDR= was not found in the library.
LOC ARGUMENT ERROR.	CPMEM	The first word address or last word address parameter specified on the LOC control statement was nonnumeric.
LOC RANGE ERROR.	CPMEM	Either the first word address was greater than the last word address or the last word address was greater than the user's field length.
LOCAL FILE LIMIT, AT nnn.	PFM	The job's local file limit has been exceeded by an attempt to GET or ATTACH the file.
LOCAL FILE LIMIT, FILE fff AT nnn.	CIO	The job's local file limit was exceeded in an attempt to define another file or attach an existing file to the job.
LPxx,... .		Refer to the EQxx,... series of corresponding messages for full descriptions of messages beginning with LPxx,... .
LQxx,... .		Refer to the EQxx,... series of corresponding messages for full descriptions of messages beginning with LQxx,... .
LRxx,... .		Refer to the EQxx,... series of corresponding messages for full descriptions of messages beginning with LRxx,... .

Message	Routine	Description
MASS STORAGE DIRECTORY NOT WRITTEN.		On a GTR control statement, user requested that a mass storage directory record be written on a nonmass storage file.
MASTER USER NUMBER REQUIRED.	PROFILE	Master user number must be present in control point area (set via USER control statement) for a master user list and for an enquire from other than system origin or special accounting user.
MDxx, Cyy, ec, ann, Stttt, FNqqqq-r. or MDxx, Cyy, ec, ann, Stttt, Ux Cxxxx Stttt.	6MD	An error has been detected on mass storage device xx. The nature of the error is determined by examining each parameter in the message. xx EST ordinal of 841 disk yy Channel number ec Error code (one of the following): PE Parity error/checkword error AD Address error ST Device status error FN Function reject for any device connected to data channel converter (6681) or function timed out with no response RS Device reserved NR Device not ready a Type of operation (one of the following): R Read W Write nn Retry count; error is considered irrecoverable after the following number of retries. PE 10 AD 10 ST 64 FN 10 RS Indefinite NR Indefinite tttt Device status; implies there was an incomplete transfer if status does not indicate an error qqqq Function rejected r Data channel converter (6681) status, if present Ux Physical unit } Cxxxx Upper address } Physical address Sxxxx Lower address }
MEMORY OVERFLOW.		Insufficient storage was allowed for an OPLEDIT run.
MESSAGE LIMIT.	1AJ	The number of messages issued by the job 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.
MFL REQUEST TOO SMALL, MINIMUM USED.	CONTROL	MFL request was less than CONTROL's RFL= value. CONTROL's RFL= value is used for this MFL request, thus allowing further MFL requests.
MIXED CHARACTER SET OPL.		Records of more than one character set were encountered on the old program library.
MONITOR CALL ERROR	1AJ	RA+1 call unrecognized.
MT... .		Refer to the EQ,... series of corresponding messages for full description of messages beginning with MT.

Message	Routine	Description
M. T. NOT AVAILABLE ON FILE fff AT nnn.	CIO	The magnetic tape executive is not executing.
MT/NT CONFLICT	RESEX	Conflict exists between 7-track and 9-track tape descriptors. For example, a request for a 9-track tape specifies 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, or if dt=NT and bit 56 is not set, the message is issued.
MULTI-FILE NAME NOT FOUND fff AT nnn.	CIO	User issued a *POSMF* on a nonexistent file on a multifile tape.
NO CONNECT TIME AVAILABLE.	CHARGE	The user has accumulated the maximum connect time allowed for the specified project number.
NO CPU TIME AVAILABLE.	CHARGE	The user has accumulated the maximum CPU time allowed for the specified project number.
NO DIRECTIVES.		Directive file was empty.
NO EOR FOUND ON ZZZZZDF.		Illegal file format for ZZZZZDF. The FILE control statement (described in the Record Manager Reference Manual) is used to update the file information table (FIT) which is required for files the Record Manager accesses. The system uses information the programmer supplies on the FILE statement to prestore FIT information in file ZZZZZDF.
NO INPUT FILE FOUND.	QFM	No valid input file exists; functions cannot be performed.
NO LINE NUMBER ON SORT FILE.	SORT	A line on the input file to a SORT request is missing a line number or a line exceeded the 150-character limit.
NO LINE TERMINATOR.	COPYC	A copy operation was attempted on a line longer than 150 characters which did not contain a line terminator.
NON-MATCHING CONVERSION		The conversion mode required for the tape is not the same as that specified on the control statement. This is only a warning message.
NO READ FILE - lfn.	SUBMIT	The specified file cannot be found.
filename NOT DECLARED RANDOM.	LIBEDIT	An EOF was encountered on the nonrandom file, filename.
lfn NOT FOUND.	RESTART	RESTART was unable to retrieve a file named, but not included, on lfn.
NORERUN/RERUN IGNORED FROM TTY JOBS.	QFM/CONTROL	User entered NORERUN/RERUN from a terminal. The command is ignored.
xxx NOT IN PP LIB.	SFP	PP package xxx was not found in PP libraries.

Message	Routine	Description
xxx NOT IN PP LIB. - CALLED BY yyy.	SFP	PP package xxx was not found in the PP libraries and was called by package yyy.
pfn NOT FOUND, AT nnn.	PFM	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.
lfn NOT ON MASS STORAGE, AT nnn.	PFM	The file to be saved is not on mass storage; the first track of the file is not recognizable.
NT.... .		Refer to the EQ,... series of corresponding messages for full description of messages beginning with NT.
NO WRITE ENABLE, ON fff AT nnn.	1MT	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 669) does not support 200-bpi density.
OLDPL ERROR.	UPDATE	Update program library format was bad.
OPERATOR DROP.	3AB	The job was dropped by the operator.
OPLEDIT COMPLETE.	OPLEDIT	Informative message indicating OPLEDIT completion.
OPLEDIT ERRORS.	OPLEDIT	Errors were encountered while modifying a particular deck.
OUTPUT FILE LIMIT	LFM	The total number of files disposed to the output queue by the job has exceeded the limit for which the user is validated.
OUTPUT FILE LIMIT, FILE fff AT nnn.	CIO	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.
OVERLAPPING INSERT OR DELETE.		Insertions and deletions affect the same deck.
OVERLAY FILE EMPTY.	3AD	No data appears in the requested file.
OVERLAY FILE NOT FOUND.	3AD	The specified file was not available.

Message	Routine	Description
OVERLAY NOT FOUND.	3AD/3AE	The specified overlay was not found.
OVERLAY NOT FOUND IN LIBRARY.	LDR	The specified overlay was not found in the system library.
PACK PARAMETER ERROR.	PACK	The PACK control statement contains an error.
PARITY ERROR - RESTARTED FROM kk.	RESTART	Because RESTART detected a parity error in attempting to restart from the specified checkpoint nn, the alternate checkpoint kk was used instead.
PBC ARGUMENT ERROR.	CPMEM	Either the first word address or the last word address specified on a PBC control statement was nonnumeric.
PBC FWA .GT. LWA.	CPMEM	The first word address was greater than the last word address.
PBC RANGE ERROR.	CPMEM	The last word address parameter specified on a PBC statement was greater than or equal to the user's field length.
PERMIT LIMIT EXCEEDED, AT nnn.	PFM	Permit limit for private file has been exceeded.
PFM ABORTED, AT nnn.	PFM	Error flag detected at PFM control point.
PFM ILLEGAL REQUEST, AT nnn.	PFM	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 attempted on a public file
PF UTILITY ACTIVE, AT nnn.	PFM	Because a permanent file utility is currently active, the operation was not attempted; the user should retry the operation.
PL ERROR IN DECK dname.	MODIFY	Error encountered in processing deck dname.
POSITION ERROR ON--xxxxxxx.		File xxxxxxx was not repositioned after being checkpointed because CHKPT detected an address error.
POSITION LOST, fff AT nnn.	1MT	During write error recovery, the system could not find the last good block of data, making it impossible to successfully perform error recovery.
PP CALL ERROR.	3AB	The monitor detected an error in a CPU request for PP action.
PROFILE ABORTED.	PROFILE	Error flag is set at control point.
PROFILE FILE CREATE COMPLETE.	PROFILE	Creation run is complete.
PROFILE FILE DATA BASE ERROR.	PROFILE	Project file does not contain level 0 and level 1 blocks.
PROFILE FILE INQUIRY COMPLETE.	PROFILE	Enquire run is complete.
PROFILE FILE LIST COMPLETE.	PROFILE	List run is complete.
PROFILE FILE REFORMAT COMPLETE.	PROFILE	Reformat run is complete.
PROFILE FILE SOURCE COMPLETE.	PROFILE	Source run is complete.
PROFILE FILE UPDATE COMPLETE.	PROFILE	Update run is complete.

Message	Routine	Description
PROGRAM FILE EMPTY.	TCS	A load of an empty data file was attempted.
PROGRAM LIBRARY EMPTY.		The old program library contained no data.
PROGRAM NOT FOUND.	EXU	The program to be loaded was not found on the specified library file.
PROGRAM NOT ON MASS STORAGE.	EXU	The program does not reside on a mass storage device.
PROGRAM STOP AT xxxxxx.	3AB	The monitor detected a program stop instruction at address xxxxxx.
PROGRAM TOO LONG	EXU	The program does not fit in the available storage.
PROTECTED FILE		The user has attempted to release a locked file.
PRU LIMIT, AT nnn.	PFM	The job's mass storage PRU limit was exceeded during preparation of a local copy of an indirect access file.
PRU LIMIT, FILE fff AT nnn.	CIO	The job's mass storage PRU limit was exceeded during an attempt to write or extend this file.
PRUS REQUESTED UNAVAILABLE.	3PF	The number of PRUs requested is not available.
PRUS REQUESTED NOT AVAILABLE, AT nnn.	PFM	The number of PRUs specified via the S parameter on the DEFINE request is not available.
QFM ARGUMENT ERROR.	QFM	One of the following: ● Address is outside field length ● Address is equal to 1 ● Origin code is out of range ● ID code is out of range
QFM EOI BAD ON ATTACHED FILE.	QFM	The EOI sector cannot be found on the specified file.
QFM FILE ALREADY ATTACHED.	QFM	The specified file is already attached to the control point.
QFM FILE EMPTY.	QFM	The submitted file has not been used.
QFM - FILE IGNORED filename.	QFM	The file was ignored because it had an illegal origin or type code. It could indicate a bad IQFT file.
QFM FILE NAME ERROR.	QFM	The lfn specified does not check as a valid file name.
QFM FILE NOT FOUND.	QFM	The submitted file could not be found.
QFM FILE NOT ON MASS STORAGE.	QFM	The submitted file does not reside on mass storage.
QFM ILLEGAL EQUIPMENT.	QFM	The equipment specified in FET+7 either is not mass storage or is not in the range of the EST.
QFM ILLEGAL FILE TYPE.	QFM	The submitted file is not a local file.
QFM ILLEGAL ID CODE.	QFM	The ID code is out of range.
QFM ILLEGAL ORIGIN TYPE.	QFM	The origin type for the submitted file is not batch or Export/Import.

Message	Routine	Description
QFM ILLEGAL REQUEST.	QFM	One of the following: ● Specified function illegal or undefined ● Job did not have SSJ= entry point ● Auto recall bit was not set
QFM INTERLOCK ERROR.	QFM	Track interlock could not be set due to a conflict.
QFM TRACK MISMATCH.	QFM	The file about to be purged is not the same file that was previously attached. The first track in the FST does not equal the one from the DULL word.
QFM UNABLE TO INTERLOCK MST.	QFM	Informative message.
QUEUE FILE UTILITY COMPLETE.	QFSP	Informative message.
QUEUED FILES LOST.	QREC	Files which process error conditions were not requeued. This error should never occur but may if QREC was aborted and could not modify its files correctly. A level 0 deadstart will recover the queues.
RA.SSC OUT OF RANGE.	1MA	The subsystem receiving the buffer pointer (RA.SSC) word has one or more fields outside the subsystem field length.
RANDOM ADDRESS NOT ON FILE fff AT nnn.	CIO	The random address specified was not within the bounds of the file. The system provides a dump of the FET on file OUTPUT.
READ AFTER WRITE, fff AT nnn.	1MT	The user attempted to read a tape on which the last operation was a write.
READ FILE BUSY - lfn	SUBMIT	The read file is found to be busy (direct access file only).
nnnnn RECORDS CONVERTED.		Informative message indicating number of records (nnnnn) converted from one character set to another.
n RECORD(S) NOT REPLACED.	LIBEDIT	Informative message; the job is aborted unless the D option was specified.
RECORD SIZE EXCEEDS 500.		The maximum line length for a record to be converted (500 characters) was exceeded.
RECORD TOO LONG.	CPMEM	The record is too long for available memory. Available memory is filled and the excess data is skipped. In response to a WBR request, the record length parameter was greater than or equal to the user's field length.
REPRIEVE IMPOSSIBLE - BAD CHECKSUM	SFP	Postrecovery checksum does not match prerecovery checksum.
REQUEST UNDEFINED ON DEVICE fff AT nnn.	CIO	The specified function cannot be performed on the device on which the file resides. The system provides a dump on the FET on file OUTPUT.
REQUESTED FL BEYOND MFL	1MA	The job's memory request has exceeded the maximum field length for a job step.
RERUN NOT POSSIBLE.	IDS	Operator attempted to rerun a job that is in no-rerun mode.
RESEX DETECTED ERROR.	LFM	The resource executive (RESEX) detected an error.

Message	Routine	Description
RESEX FAILURE, AT nnn.	PFM	The resource executive (RESEX) has detected a fatal error.
RESOURCE DEMAND ERROR.	RESEX	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.
RESOURCE TYPE ERROR.	RESEX	The user specified an illegal resource type.
jobname RESTARTED FROM yy/mm/dd. hh.mm.ss.	RESTART	The checkpointed job identified by jobname was restarted from the checkpoint taken on the specified data and time. This message is issued whenever a checkpoint job is restarted.
RFL BEYOND MFL.	CPM	The RFL request is greater than the maximum field length for a job step.
ROLLIN FILE BAD.	1RI	An illegal format was detected in the roll-in file.
ROUTE CONTROL CARD ERROR.	ROUTE	Format of the control statement is incorrect.
ROUTE *DC* INCOMPATIBLE WITH *EC*.	ROUTE	The user specified a DC/EC combination that is not legal. If the DC parameter implies a print file, the EC parameter must be for print files.
ROUTE ILLEGAL KEYWORD.	ROUTE	Control statement contains an illegal keyword.
ROUTE ILLEGAL *OT* PARAMETER.	ROUTE	The origin type specified by the OT parameter is illegal.
ROUTE *OT* NOT ALLOWED.	ROUTE	The user program is not system origin. Only system origin jobs can use the OT parameter.
ROUTE *REP* GT 31. DEFAULT USED.	ROUTE	The repeat count specified was greater than 31; it has been set to 0. This condition will not abort the program.
ROUTE *TID* AND *FM/UN* CONFLICT.	ROUTE	The TID parameter was specified with either the FM or UN parameter. Either one of these parameters is mutually exclusive with TID.
ROUTE *TID/FM/UN* and *ID* CONFLICT.	ROUTE	The ID parameter was specified with the TID or FM or UN parameter.
ROUTE *FID* IGNORED.	ROUTE	Informative message listed for NOS/BE compatibility.
ROUTE *PRI* IGNORED.	ROUTE	Informative message listed for NOS/BE compatibility.
ROUTE *ST* IGNORED.	ROUTE	Informative message listed for NOS/BE compatibility.
ROUTE *TID=xx - VALUE IGNORED.*	ROUTE	Informative message listed for NOS/BE compatibility.
ROUTE COMPLETE.		Issued when route is complete.
ROUTE COMPLETE. JOB NAME IS FILnam.		Issued when route is complete.

Message	Routine	Description
SECURE MEMORY, DUMP DISABLED.	1AJ	An attempt was made to dump memory protected by the system.
SFM ARGUMENT ERROR.	SFM	The argument passed to SFM was out of bounds or the FET specified did not specify a buffer of at least 100_8 words.
SFM DAYFILE BUSY.	SFM	Action was requested on a busy dayfile.
SFM ILLEGAL DAYFILE CODE.	SFM	The dayfile code passed in the FET was not within range.
SFM ILLEGAL REQUEST.	SFM	The requested function or origin type specified in the function call was not recognizable or SFM request was made and the auto recall bit was not set.
SFM TRACK INTERLOCK ERROR.	SFM	Track was either interlocked when it should not have been or not interlocked when it should have been.
SL NOT VALIDATED.	CPM	The SRU limit requested exceeds that for which the user is validated.
SMF UNABLE TO INTERLOCK DEVICE.	SFM	SFM request was not performed because the selected device could not be interlocked.
SFP CALL ERROR.	SFP	SFP was not loaded by default.
SFP/RPU UNABLE TO RESET, NOT REPRIEVED.	SFP	An attempt was made to reset when the job had not been reprieved.
SFP.xxx ILLEGAL ORIGIN CODE.	SFP	Function illegal for user's job origin.
SFP/xxx PARAMETER ERROR.	SFP	Parameter address outside FL.
SPCW CALL ERROR.	1AJ	A DMP= type call was made, and the program called is either not in the CLG or does not have a DMP= entry point defined.
SPECIAL REQUEST PROCESSING ERROR.	SFP	The SPCW word was busy.
STATUS ERROR, fff AT nnn.	1MT	An irrecoverable error was encountered. A second message line describes the error in more detail. CRC ERROR. An error was detected in cyclic redundancy character. DATA TIMING PROBLEMS. Hardware malfunctions. Another unit should be tried. 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. FLAG BIT ERROR. The tape being read with ASCII conversion contains characters not included in the 128-character set. MEMORY PARITY ERROR. A parity error was detected in the conversion memory of the tape controller. 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.

Message	Routine	Description
		POSTAMBLE ERROR. A missing or defective postamble was detected at 1600 cps. 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. IBG NOT FOUND - POSITION UNCERTAIN. False read end-of-operation occurred, and the IBG could not be located within 100 inches. Further positioning is uncertain.
SUBSYSTEM ABORTED.	3AB/1AJ	The user job was connected (either long term connection or wait response set) to a subsystem which aborted.
SYSTEM ABORT.	1AJ	Possible errors include detection of a bad rollout file by 1RI, an unrecognizable error flag, an SSJ= block outside a field length, or an invalid USER statement.
SYSTEM SECTOR ERROR.	QFM	An error occurred while reading the system sector.
TABLE OVERFLOW. JOB ABORTED.		Resubmit job with increased field length.
TAPE BLOCK DEFINITION ERROR.	RESEX	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.
TAPE FORMAT PROBABLY WRONG.	1MT	This message is issued in addition to one of the following messages. BLOCK SEQUENCE ERROR, fff AT nnn. BLOCK TOO LARGE, fff AT nnn. WRONG PARITY, fff AT nnn. if one of these error conditions occurs on the first block.
TIME LIMIT.	3AB	The monitor detected that the time limit for the job step has expired.
TL NOT VALIDATED.	ACCFAM	The time limits specified on the job statement exceed that for which the user is validated.
	CPM	The time limit requested exceeds that for which the user is validated.

Message	Routine	Description
TOO MANY ARGUMENTS.	TCS	The number of arguments on the control statement exceeds that allowed by the program.
TOO MANY ARGUMENTS.	COPYB	More arguments were specified on a copy request than are allowed on that statement.
TOO MANY DEFERRED BATCH JOBS.	QFM	The user is not validated for this function or he has more jobs in the system than he is allowed. (All jobs in batch queues and E/I 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.
TRACK ALREADY ASSIGNED	QFM	The track byte for the IQFT file in the DULL word in the MST is already assigned.
TRACK LIMIT, FILE fff AT nnn.	CIO	The device on which the file resides is full.
UNABLE TO READ IQFT FILE.	IMS/MSI	An attempt to initialize inactive queues failed because the IQFT file could not be read.
UNIDENTIFIED PROGRAM FORMAT.	3AE	The file the user requested to be loaded was not in a recognizable format.
UNRECOVERABLE MS ERROR.	QFM	An irrecoverable mass storage error was detected during an I/O operation.
UPMOD COMPLETE.	UPMOD	Informative message indicating UPMOD completion.
VERIFY ERRORS.	VERIFY	Errors were encountered during VERIFY routine.
WBR ARGUMENT ERROR.	CPMEM	The record length parameter specified on a WBR statement was nonnumeric.
WRITE ON READ-ONLY FILE fff AT nnn.	CIO	Either the user attempted to write on a file with write interlock or the direct access file was not attached in WRITE mode.
WRITE OVER LABEL ILLEGAL ON fff AT nnn.	1MT	The user is not allowed to destroy the VOL1 label.
WRONG PARITY, fff AT nnn.	1MT	A 7-track tape is being read in opposite parity from which it was written.
25555 FIELD LENGTH INCREASE.	LIBEDIT	The job field length was too small for LIBEDIT. Field length was increased to 26K.

LFM ERROR CODES

The following octal error codes are returned to the error code field of the FET word 0, bits 10 through 13 in response to LFM requests.

Error Codes	Description
1	File not found
2	File name error
3	Illegal file type
4	File empty
6	Duplicate common file name
7	Illegal equipment
10	Equipment not available
11	Duplicate file name
12	Illegal user access
13	Illegal user number
14	Illegal ID code
15	Resource executive (RESEX) detected an error
16	I/O sequence error
17	Output file limit
20	Local file limit
21	No mass storage available
22	Illegal file mode
23	FET too short
24	GETFNT table too large
25	Illegal change in file/origin type

PFM ERROR CODES

The following error codes are returned to the error code field of the FET word 0, bits 17 through 10 in response to PFM requests.

Error Codes — Description

1 — The specified direct access file is attached in the opposite mode.

2 — One of the following:

- The specified permanent file could not be found.
- The specified account 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 macro, the specified local file is not attached to the control point, is a direct access file, or is an execute-only file.

3 — The file specified on a SAVE macro contains no data.

4 — The file to be saved is not on mass storage; the first track of the file is not recognizable.

5 — The user has already saved or defined a file with the name specified.

6 — The user attempted to define a file that was not a local file.

7 — File name contains illegal characters.

10 — The user is not validated to create direct access or indirect access files or to access auxiliary devices.

11 — 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, this message is issued.

Error Codes — Description

12 The local file specified for a SAVE, REPLACE, or APPEND command exceeds the length allowed, or the direct access file specified for an ATTACH in WRITE, MODIFY, or APPEND mode exceeds the direct access file length limit for which the user is validated.

13 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 attempted on a library file

14 Access to the permanent file device requested is not possible.

15 The device on which the file resides may not contain direct access files because:

1. The device is not specified as a direct access device in the catalog descriptor table.
2. The device is not specified as ON and initialized in the catalog descriptor table.
3. The device is a dedicated indirect access permanent file device.

16 Because a permanent file utility is currently active, the operation was not attempted; the user should retry the operation.

17 An error occurred in a read operation during a file transfer.

20 The number of files in the user's catalog exceeds the limit (refer to LIMITS control statement, section 6).

21 The cumulative size of the indirect access files in the user's catalog exceeds the limit (refer to LIMITS control statement, section 6).

Error Codes — Description

22 The number of PRUs specified via the S parameter on the DEFINE macro is not available.

23 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.

24 The job's local file limit has been exceeded by an attempt to GET or ATTACH the file.

25 The job's mass storage PRU limit has been exceeded during preparation of a local copy of an indirect access file.

26 Permit limit has been exceeded for a private file.

30 The resource executive has detected a fatal error.

31 No allocatable tracks remain on equipment xx, where xx is the EST ordinal.

32 The length of a file does not equal the catalog length; the action taken depends on the type of command issued.

Command	Action
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.

33 Permit random address error.

34 The system sector data for the file does not match the catalog data.

35 The same file was found twice during a catalog search. This error can occur for APPEND or REPLACE commands after a file is found and purged and the catalog search is continued.

36 Error flag detected at PFM control point.

37 An error was encountered in reading a portion of the permanent file catalog or permit information.

Table 1-B-1 specifies the action PFM takes if it detects an error while reading mass storage. The symbols used in the table designate the type of response PFM makes and are defined as follows:

Symbol	Description	Code
DTE	DATA TRANSFER ERROR.	17
EOI	Processing continues as if an EOI was encountered.	
MSE	MASS STORAGE ERROR.	37
FNF	pfn NOT FOUND.	2
DAF	DIRECT ACCESS FILE ERROR.	34
FLE	FILE LENGTH ERROR.	32

TABLE 1-B-1. PERMANENT FILE ERROR CONDITIONS

	Command								
Activity	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.

LIBEDIT C

LIBEDIT is a binary record management program that is used to:

- Create and maintain a program library file
- Copy records to a program library file
- Delete records from a program library file
- Replace records on a program library file

Binary logical records are the basic unit manipulated. LIBEDIT manipulates the records of the old program library file and optional replacement files. Records for replacement can be on one or more secondary files. Replacement is the implicit mode of a LIBEDIT run. Additions and no-replacements must be explicitly requested.

LIBEDIT manipulates the following record types.

- Relocatable central processor program (REL)
- Central processor overlay (OVL)
- Multiple entry point overlay (ABS)
- 6000 peripheral processor program (PP)
- 7600 peripheral processor program (PPU)
- Modify old program library deck (OPL)
- Modify old program library common deck (OPLC)
- Modify old program library directory (OPLD)
- User library programs (ULIB)
- Unrecognizable as a program (TEXT)
- CAP capsule loader record (Supported by CDC CYBER Loader 1.3)

Formats are further described in appendix G, volume 2.

LIBEDIT executes in two phases. During the first phase, it reads directives and replacement records. It groups directives by type and file and groups corrections when several insertions take place relative to the same record.

During the second phase, LIBEDIT performs modifications and generates the new program library. 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 the conflicting directives (or a simulated form of the directives), issues an error message, and aborts the job. If the D option was specified, LIBEDIT continues processing the directives.

CONTROL STATEMENT FORMAT

The following control statements call the LIBEDIT program to be loaded and executed. Parameters specify mode and files.

LIBEDIT(p_1,p_2,...,p_n)

The optional parameters, p_i, can be in any order within the parentheses. Generally, a parameter can be omitted or can be in one of the following forms.

a (C, R, and V only)

a=lfn

a=0

a is one of the following options: I, P, N, L, LO, B, C, R, and V. lfn is the 1- to 7-alphanumeric character file name. LIBEDIT accepts only one instance of any parameter.

Option	Description
I=lfn	Directives comprise the next record on file lfn
I=0	No directive input
I omitted	Directives are on file INPUT
P=lfn	File lfn contains the old program library
P=0	No old program library file
P omitted	Old program library is on file OLD
N=lfn	New program library will be written on file lfn
N=0	Illegal; no error message is issued, if used
N omitted	New program library will be written on file NEW
	NOTE The new program library is evicted prior to processing (refer to EVICT statement, section 7).
L=1	Short correction listing (includes only directives, modifications, and errors) on the file specified by the LO parameter
L=0	No output is listed
L omitted	Full correction listing is written on the file specified by the LO parameter
LO=lfn	List output on file lfn
LO omitted	List output on file OUTPUT
B=lfn	Use file lfn for the replacement file
B=0	Do not use a default replacement file
B omitted	Use file LGO as the default replacement file
C	Copy the new library file over the old library file after processing
C omitted	Do not copy the new library file over the old library file after processing
R	Do not rewind library files after processing
R omitted	Rewind old and new library files after LIBEDIT and VFYLIB processing
V	Call VFYLIB after LIBEDIT processing
V omitted	Do not call VFYLIB to verify libraries after LIBEDIT processing
D	Ignore errors and continue
D omitted	Do not ignore errors; abort job

LIBEDIT DIRECTIVES

Directives comprise a program record on file INPUT or on the file specified through the I mode parameter on the LIBEDIT control statement. Directives control the record management process. A directive begins with an asterisk in column 1 followed immediately by the statement identifier. The statement identifier is delimited by a comma and/or one or more spaces. Parameters are delimited by -, a blank, an end-of-line, or a comma.

Statement parameters have no embedded blanks. If a directive does not begin with an asterisk and a statement identifier, LIBEDIT assumes the operation is a continuation of the last directive operation. If the statement was not preceded by a directive, the operation is assumed to be:

*BEFORE *, gid_1, gid_2, . . . , gid_n.

Note, however, that gid entries cannot be split between statements. For example, the statements

```
*B,OVL/P1,OVL/P2,...,OVL/P
N
```

do not constitute a valid directive. The last entry would not be processed as OVL/PN. On the other hand, the statements

```
*B,OVL/P1,OVL/P2
OVL/P3
OVL/PN
0
TEXT/T1
```

do constitute a valid directive and would be processed in the same manner as:

```
*B,OVL/P1,OVL/P2,OVL/P3,OVL/PN,0,TEXT/T1
```

Directives are not required. If they are not provided, LIBEDIT replaces the records of the old program library file that have the same name and type as the records on the correction file, and LIBEDIT writes the new library.

Parameters common to many of the correction directives are the reference record identifier (rid) and the group record identifier (gid).

rid — The rid parameter specifies a reference point for a correction. It can be in one of the following forms.

- type/rname — Reference record is of the specified type
- rname — Reference record is the implied type (refer to type)
- * — Reference point is an end-of-file mark (*BEFORE card only)

gid — One or more gid parameters on a directive indicate records or groups of records to be inserted, deleted, or replaced. A gid can be in one of the following forms.

- type/rname — Single record of the specified type
- $type_1/rname_1$-$type_2/rname_2$ — Group of records beginning with rname of $type_1$ and ending with $rname_2$ of $type_2$. Types are specified or implied.

  rname — Record identifier can be one of the following.

  rname — Name of record

* If used for rname_1 on an INSERT, AFTER, BEFORE, or IGNORE, an * indicates that all records on the library of the specified or implied type are to be inserted or ignored.

If used for rname_2 on INSERT, AFTER, BEFORE, or IGNORE, an * indicates that all records of type_1, starting with rname_1, are to be inserted or ignored.

0 Indicates that a zero-length record is to be inserted.

type Identifies the type of the named record. When type is absent from a rid or gid parameter, LIBEDIT uses the type most recently specified on a directive. For valid types, refer to the description of the TYPE directive.

LIBEDIT recognizes the following directives.

Directive	Definition
*ADD	Adds records at end of library.
*BEFORE or *B	Inserts records before the named record.
*BUILD	Builds an index at end of new file.
*COMMENT	Adds comment to prefix table.
*COPY	Copies new file to old at end of editing.
*DATE	Adds date and comment to prefix table.
*DELETE or *D	Deletes specified records.
*FILE	Declares additional correction file.
*IGNORE	Ignores records when reading correction file.
*INSERT or *I or *AFTER or *A	Inserts records from correction file after named record.
*NOREP	Does not automatically replace records from named file.
*RENAME	Renames record.
*REPLACE	Replaces records on old file with records from correction file. Optionally declares current correction file as no-replace.
*REWIND	Designates file to be rewound before and after editing.
*TYPE or NAME	Sets type of library to be used for default.

FILE

The directive format is:

*FILE lfn

lfn Name of the additional replacement file; subject to operating system restrictions on file names. If lfn is an *, LIBEDIT uses the replacement file specified by the LIBEDIT statement or the default file (LGO), if none is specified.

The FILE directive declares a secondary file as an additional file that contains replacement records. LIBEDIT directives following a FILE statement specify records on the declared replacement file.

REWIND

The directive format is:

*REWIND lfn

lfn Name of file to be rewound

LIBEDIT rewinds the specified file before and after editing.

TYPE OR NAME

The formats for the directives are:

*TYPE type

*NAME type

type Specifies default type of internal record format:

ABS	Multiple entry point overlay
CAP	Capsule loader record (Supported by CDC CYBER Loader 1.3)
OPL	Modify old program library deck
OPLC	Modify old program library common deck
OPLD	Modify old program library directory
OVL	CPU overlay
PP	6000 series format peripheral processor unit program
PPU	7600 format peripheral processor unit program
REL	Relocatable CPU program
TEXT	Unrecognizable as a program
ULIB	User library program; begins with a ULIB type record and terminates with OPLD type record

Any explicit use of a type or a rid or gid parameter resets the default value to the new type.

With the TYPE (or NAME) directive, the user specifies the type of record to which subsequent LIBEDIT directives refer. A type specification is in effect until the next TYPE (or NAME) directive is supplied or until a type is explicitly declared on another directive. If no TYPE or NAME directive is supplied or no explicit type is used, the type is TEXT.

For example:

```
*BEFORE *, JANE-*
*DELETE HENRY-IDA
*INSERT GEORGE,MARY
*TYPE REL
```

is equivalent to

```
*BEFORE *, REL/JANE-*
*DELETE REL/HENRY-REL/IDA
*INSERT REL/GEORGE,REL/MARY
```

INSERT OR AFTER

The formats for the directives are:

*INSERT rid, gid_1, gid_2, . . . , gid_n	or	*I rid, gid_1, gid_2, . . . , gid_n
*AFTER rid, gid_1, gid_2, . . . , gid_n	or	*A rid, gid_1, gid_2, . . . , gid_n

rid — Identifies the record on the old library file after which the specified records or groups of records are to be inserted

gid_1 — Identifies the records or groups of records from the replacement file to be inserted after rid

An INSERT or AFTER directive directs LIBEDIT to insert records or groups of records from the current replacement file after the specified old library record for transcription to the new library file. The current replacement file is the most recent file specified by a FILE directive or by the LIBEDIT control statement. Insertion of records causes automatic deletion of the old records having the same names and types from the old library file.

An example of the use of this directive is:

```
*INSERT OPL/LEA,TEXT/OSCAR-*
```

These statements direct LIBEDIT to insert, after the OPL deck LEA on the old library file, all TEXT records from OSCAR until an end-of-file mark is encountered. If any of these TEXT records have the same name as a TEXT record that is already on the old library file, the old TEXT record is not transcribed to the new library file.

BEFORE

The directive formats are:

*BEFORE rid, gid_1, gid_2, . . . , gid_n or *B rid, gid_1, gid_2, . . . , gid_n

rid — Identifies the record on the old library file before which the specified records are to be inserted. On the form omitting the directive name, rid is assumed to be * (that is, insert before end-of-file).

gid_i — Identifies records or groups of records from the replacement file to be inserted before rid.

A BEFORE directive causes LIBEDIT to insert records or groups of records from the current replacement file before the specified old library record for transcription to the new library file. The current replacement file is the most recent replacement file specified by a FILE directive or by the LIBEDIT control statement. Insertion of records causes automatic deletion of the old records having the same names and types from the old library file.

DELETE

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 to be deleted from the old library file. An asterisk cannot be used.

The DELETE directive causes LIBEDIT to suppress copying of the specified records from the old library file to the new library file.

An example of the use of this directive is:

```
    *DELETE PPU/LAD-REL/RUN
```

This statement directs LIBEDIT to delete records starting with 7600 PPU program LAD through relocatable CPU program RUN.

IGNORE

The directive format is:

*IGNORE gid_1, gid_2, . . . , gid_n

gid_i — Identifies records or groups of records from the replacement file to be ignored.

The IGNORE directive causes LIBEDIT to ignore a record or group of records on the current replacement file during record processing.

An example of the use of this directive is:

```
        *IGNORE FRAN-*
    *FILE WOMAN
```

LIBEDIT ignores program FRAN of the current type and all following programs of the current type until an end-of-file mark on the replacement file WOMAN is encountered.

ADD

The directive format is:

*ADD lib, gid_1, gid_2, . . . , gid_n

lib — Specifies that the library is to be added to the old program library file before the zero-length record for the old library file indicated. A library cannot be added if there is no zero-length record.

LIB1 to LIB63	Libraries 1 through 63 on the old program library file.

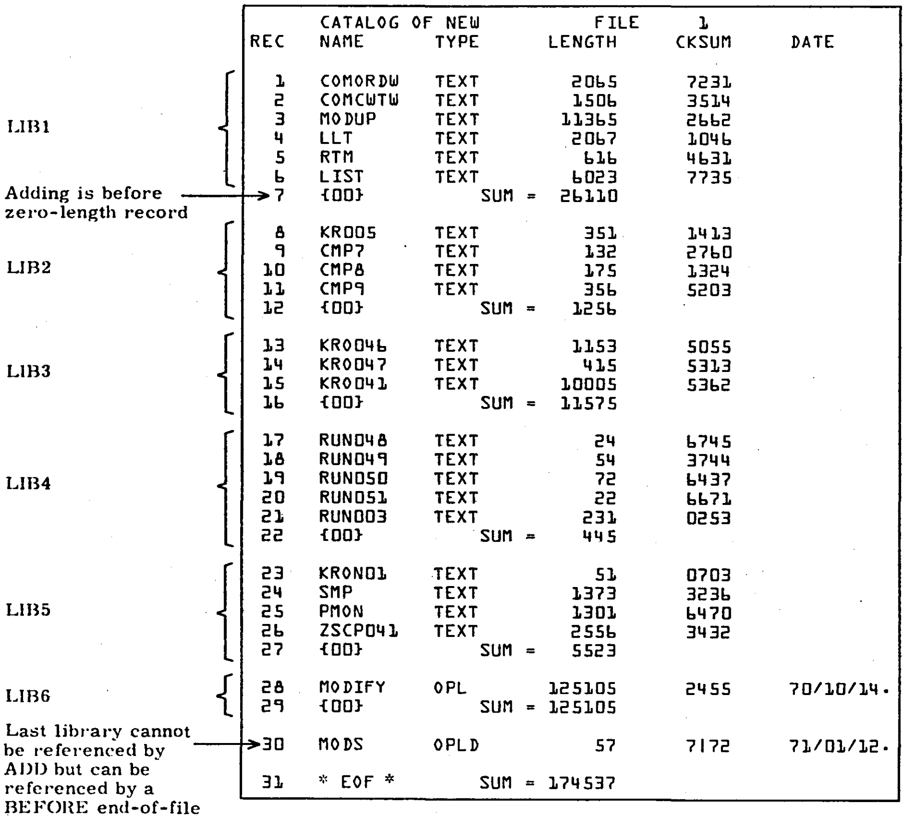
gid_i — Identifies records or groups of records to be added to the specified library.

The ADD directive causes LIBEDIT to append records to the specified library for transcription to the new library. Two libraries are separated by a zero-length record on the new library file.

NOTE

Directories are determined from file OLD; adding a zero-length record does not change the directory of the library being added.

Figure 1-C-1 illustrates where records are inserted with the ADD directive.



```
        CATALOG OF NEW          FILE      1
  REC   NAME       TYPE       LENGTH    CKSUM       DATE

    1   COMORDW    TEXT         2065     7231
    2   COMCWTW    TEXT         1506     3514
    3   MODUP      TEXT        11365     2662
    4   LLT        TEXT         2067     1046
    5   RTM        TEXT          616     4631
    6   LIST       TEXT         6023     7735
    7   {00}           SUM =   26110

    8   KR005      TEXT          351     1413
    9   CMP7       TEXT          132     2760
   10   CMP8       TEXT          175     1324
   11   CMP9       TEXT          356     5203
   12   {00}           SUM =    1256

   13   KR0046     TEXT         1153     5055
   14   KR0047     TEXT          415     5313
   15   KR0041     TEXT        10005     5362
   16   {00}           SUM =   11575

   17   RUN048     TEXT           24     6745
   18   RUN049     TEXT           54     3744
   19   RUN050     TEXT           72     6437
   20   RUN051     TEXT           22     6671
   21   RUN003     TEXT          231     0253
   22   {00}           SUM =     445

   23   KRON01     TEXT           51     0703
   24   SMP        TEXT         1373     3236
   25   PMON       TEXT         1301     6470
   26   ZSCP041    TEXT         2556     3432
   27   {00}           SUM =    5523

   28   MODIFY     OPL        125105     2455     70/10/14.
   29   {00}           SUM =  125105

   30   MODS       OPLD           57     7172     71/01/12.

   31   * EOF *        SUM =  174537
```

LIB1
Adding is before zero-length record
LIB2
LIB3
LIB4
LIB5
LIB6
Last library cannot be referenced by ADD but can be referenced by a BEFORE end-of-file

Figure 1-C-1. Adding to the Old Program Library

BUILD

The directive format is:

*BUILD dname

dname Name of directory record.

The BUILD directive requests LIBEDIT to construct and append a directory record in Modify format to the new library file. If the old library file has such a directory, LIBEDIT automatically generates a new directory deck without an explicit BUILD request.†

COMMENT

The directive format is:

*COMMENT rid comment

rid Name of the record on the replacement or old library file.

comment A string of up to 40_{10} characters that is suitable as a comment. Additional characters are truncated.

The COMMENT directive adds a comment to the prefix (77) table for a program on a replacement file or the old library file. If the program previously did not have a prefix table, LIBEDIT generates one that includes the date and the comments.

DATE

The directive format is:

*DATE rid comment

rid Name of the record on the replacement or old library file.

comment A string of up to 40_{10} characters that is suitable as a comment. Additional characters are truncated.

The DATE directive adds the current date and the specified comment to the prefix (77) table for a program on a replacement file or the old library file.

NOREP

The directive format is:

*NOREP $lfn_1, lfn_2, \ldots, lfn_n$

The NOREP directive declares the specified replacement files to be no-replace files. LIBEDIT does not replace all records of the old library file with records on the no-replace file having identical names but selectively replaces records from a no-replace file according to REPLACE, INSERT, and BEFORE directives.

† BUILD can also be used to change the directory name.

RENAME

The directive format is:

*RENAME rid,name

rid — Name of the record on the replacement or old library file to be renamed.

name — New name of the record (1 to 7 characters).

The RENAME directive assigns a new name to a record on the old library or the current replacement file for transcription to the new library file. If the renamed record is referenced by another correction statement in the same run, the old name should still be used.

REPLACE

The directive format is:

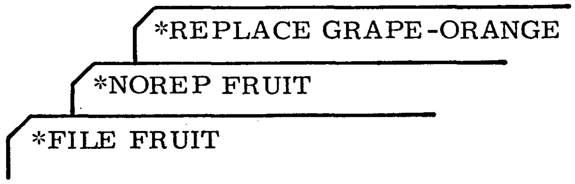
*REPLACE $gid_1, gid_2, \ldots, gid_n$

gid_i — Name of the record or record group from the replacement file to replace on the old library file.

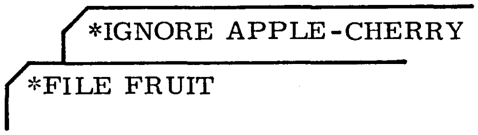
The REPLACE directive directs LIBEDIT to selectively replace records on the old library file with records of the same name from a current replacement file that has been declared a no-replace file (refer to the NOREP statement description). Thus, the user can selectively replace records by using the NOREP and REPLACE directives, or he can selectively not replace records by using the IGNORE directive according to the circumstances.

An example of the use of this directive follows: A user has a replacement file named FRUIT containing records APPLE, CHERRY, GRAPE, and ORANGE. Records having the same names are on the old library file. The user wishes to retain records APPLE and CHERRY but replace records GRAPE and ORANGE.

The following two sequences of directives produce the same results.



```
*REPLACE GRAPE-ORANGE
*NOREP FRUIT
*FILE FRUIT
```



```
*IGNORE APPLE-CHERRY
*FILE FRUIT
```

COPY

The directive format is:

*COPY

The COPY directive directs LIBEDIT to copy the new library file to the old library file after it has processed all correction statements.

LIBEDIT/LIBGEN EXAMPLES

The following examples illustrate the use of LIBEDIT and LIBGEN. LIBEDIT manipulates program library files that can contain many different record types; LIBGEN only generates a user library from relocatable (REL) records.

Example 1:

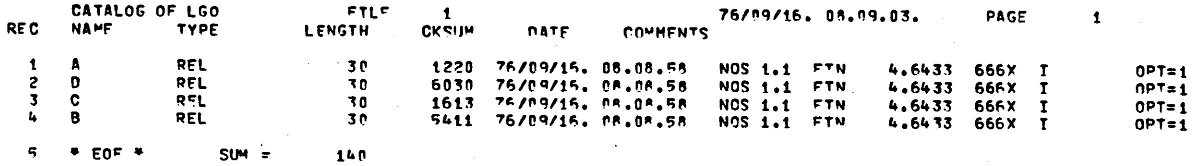
The following job builds a program library from a replacement file that consists of relocatable binary (REL) type records.

```
LIBTES1.
USER, EFD25.
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 will be written.

The first CATALOG statement gives the following listing of the LGO file.



```
        CATALOG OF LGO             FILE    1                                      76/09/16. 08.09.03.     PAGE      1
 REC    NAME         TYPE        LENGTH    CKSUM      DATE        COMMENTS

   1    A            REL             30     1220  76/09/15. 08.08.58   NOS 1.1  FTN     4.6433  666X  I          OPT=1
   2    D            REL             30     6030  76/09/15. 08.08.58   NOS 1.1  FTN     4.6433  666X  I          OPT=1
   3    C            REL             30     1613  76/09/15. 08.08.58   NOS 1.1  FTN     4.6433  666X  I          OPT=1
   4    B            REL             30     5411  76/09/15. 08.08.58   NOS 1.1  FTN     4.6433  666X  I          OPT=1

   5    * EOF *          SUM =      140
```

The P=0 in the LIBEDIT statement indicates there is no old program library. The N parameter indicates the new program library will be written on file TESTLIB. The replacement file will be the default LGO. The directives will be 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.                                    76/09/16. 08.09.03.          PAGE     1

          *BUILD LIBRARY
          *B,*,REL/A,B,C,D

 ------------------------------------------------------------------------------------------------------

      RECORDS WRITTEN ON FILE TESTLIB                             76/09/16. 08.09.03.          PAGE     2

           RECORD     TYPE       FILE       DATE       COMMENT

 INSERTED A           REL        LGO       76/09/16. 08.08.58   NOS 1.1   FTN     4.6433  666X  I           OPT=1
 INSERTED B           REL        LGO       76/09/16. 08.08.58   NOS 1.1   FTN     4.6433  666X  I           OPT=1
 INSERTED C           REL        LGO       76/09/16. 08.08.58   NOS 1.1   FTN     4.6433  666X  I           OPT=1
 INSERTED D           REL        LGO       76/09/16. 08.08.58   NOS 1.1   FTN     4.6433  666X  I           OPT=1
 ADDED    LIBRARY     OPLD       *****      76/09/16.
          **EOF**
```

The second CATALOG statement produces the following listing of information about the records on TESTLIB.

```
       CATALOG OF TESTLIB        FILE    1                          76/09/16. 08.09.04.      PAGE      1
 REC   NAME        TYPE        LENGTH    CKSUM      DATE       COMMENTS

   1   A           REL            30     1220  76/09/16. 08.08.58   NOS 1.1   FTN     4.6433  666X  I          OPT=1
   2   B           REL            30     5411  76/09/16. 08.08.58   NOS 1.1   FTN     4.6433  666X  I          OPT=1
   3   C           REL            30     1613  76/09/16. 08.08.58   NOS 1.1   FTN     4.6433  666X  I          OPT=1
   4   D           REL            30     6030  76/09/16. 08.08.58   NOS 1.1   FTN     4.6433  666X  I          OPT=1
   5   LIBRARY     OPLD           13     2073  76/09/16.

   6   * EOF *         SUM =     153
```

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).

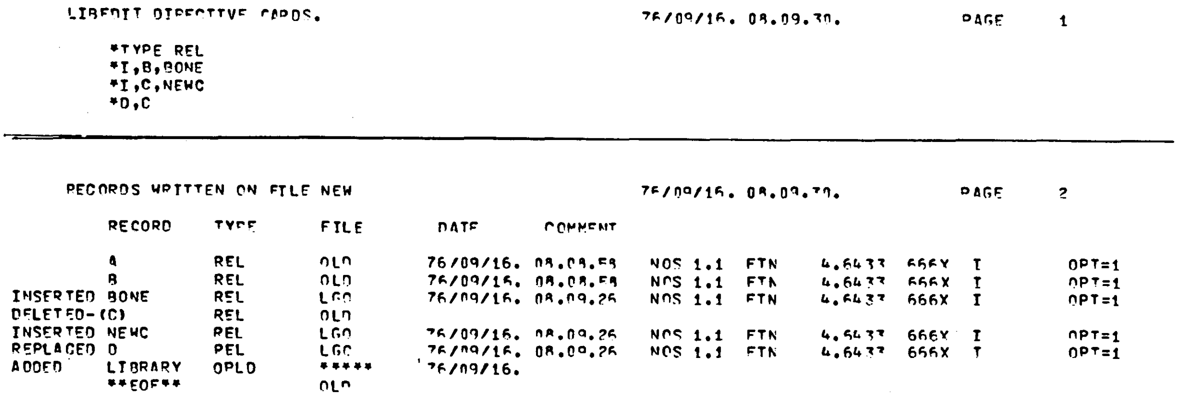
```
LIBTES2.
USER , EFD2S.
CHARGE, 16, 13N122.
FTN, L=0.
ATTACH, OLD=TESTLIB.
DEFINE, NEW=TES2LIB.
LIBEDIT.
CATALOG, NEW, R.
/EOR
                SUBROUTINE BOND
                STOP
                END
                SUBROUTINE D
                STOP
                END
                SUBROUTINE NEWC
                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 is referenced as OLD.

A direct access file (TES2LIB) is created for the new program library. This file will be 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.                                   76/09/16. 08.09.30.        PAGE     1

       *TYPE REL
       *I,B,BONE
       *I,C,NEWC
       *D,C
_____________________________________________________________________________________________________

   RECORDS WRITTEN ON FILE NEW                                76/09/16. 08.09.30.        PAGE     2

       RECORD     TYPE      FILE       DATE       COMMENT

       A          REL       OLD        76/09/16. 08.08.58  NOS 1.1  FTN   4.6433  666X  I        OPT=1
       B          REL       OLD        76/09/16. 08.08.58  NOS 1.1  FTN   4.6433  666X  I        OPT=1
INSERTED BONE     REL       LGO        76/09/16. 08.09.26  NOS 1.1  FTN   4.6433  666X  I        OPT=1
DELETED-(C)       REL       OLD
INSERTED NEWC     REL       LGO        76/09/16. 08.09.26  NOS 1.1  FTN   4.6433  666X  I        OPT=1
REPLACED D        REL       LGO        76/09/16. 08.09.26  NOS 1.1  FTN   4.6433  666X  I        OPT=1
ADDED  LIBRARY    OPLD      *****      76/09/16.
       **EOF**              OLD
```

The CATALOG shows the following content of the new program library.

```
        CATALOG OF NEW           FILE    1                                   76/09/16. 08.09.31.        PAGE       1
  REC   NAME        TYPE       LENGTH   CKSUM      DATE       COMMENTS

    1   A           REL            30    1220   76/09/16. 08.08.58    NOS 1.1   FTN      4.6433  666X  I             OPT=1
    2   B           REL            30    5411   76/09/16. 08.08.58    NOS 1.1   FTN      4.6433  666X  I             OPT=1
    3   BONE        REL            30    0677   76/09/16. 08.09.26    NOS 1.1   FTN      4.6433  666X  I             OPT=1
    4   NEWC        REL            30    1377   76/09/16. 08.09.26    NOS 1.1   FTN      4.6433  666X  I             OPT=1
    5   D           REL            30    6030   76/09/16. 08.09.26    NOS 1.1   FTN      4.6433  666X  I             OPT=1
    6   LIBRARY     OPLD           15    1312   76/09/16.

    7   * EOF *         SUM =     205
```

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's control point. 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 random 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                                   76/09/16. 08.10.04.        PAGE       1
  REC   NAME        TYPE       LENGTH   CKSUM      DATE       COMMENTS

    1   LOADLIB     ULIB           13    4257   76/09/16.
    2   A           REL            30    1220   76/09/16. 08.08.58    NOS 1.1   FTN      4.6433  666X  I             OPT=1
    3   B           REL            30    5411   76/09/15. 08.08.58    NOS 1.1   FTN      4.6433  666X  I             OPT=1
    4   BONE        REL            30    0677   76/09/16. 08.09.26    NOS 1.1   FTN      4.6433  666X  I             OPT=1
    5   NEWC        REL            30    1377   76/09/16. 08.09.26    NOS 1.1   FTN      4.6433  666X  I             OPT=1
    6   D           REL            30    6030   76/09/15. 08.09.26    NOS 1.1   FTN      4.6433  666X  I             OPT=1
    7   LOADLIB     OPLD           15    6303   76/09/16.

    8   * EOF *         SUM =     220
```

Example 4:

This job deletes a record from the user library LIBLOAD created in example 3. It does this by deleting this record from the source library (the program library TEX2LIB) from which LIBLOAD was originally generated and generating a new version of LIBLOAD.

```
LIBTES4.
USER, EFD25.
CHARGE, 16, 13N122.
ATTACH, TES2LIB.
ATTACH, LIBLOAD/M=W.
LIBEDIT, P=TES2LIB.
LIBGEN, F=NEW, P=LIBLOAD, N=LOADLIB.
CATALOG, LIBLOAD, R, U.
/EOR
*D, REL/A
/EOR
```

The program library TES2LIB (containing the relocatable routines used to generate the user library LIBLOAD) is attached to the job's control point.

The user library LIBLOAD is attached in write mode.

The LIBEDIT deletes the record A from TES2LIB. The modified new program library is written on file NEW (N parameter default). LIBEDIT produces the following listing.

```
      LIBEDIT DIRECTIVE CARDS.                             76/09/16. 08.10.30.          PAGE      1

          *D,REL/A
 ---------------------------------------------------------------------------------------------------

      RECORDS WRITTEN ON FILE NEW                          76/09/16. 08.10.30.          PAGE      2

          RECORD      TYPE      FILE        DATE        COMMENT

DELETED-(A)           REL       TES2LIB
          B           REL       TES2LIB     76/09/16. 08.08.58    NOS 1.1   FTN     4.6433  666X  T        OPT=1
          BONE        REL       TES2LIB     76/09/16. 08.09.26    NOS 1.1   FTN     4.6433  666X  T        OPT=1
          NEWC        REL       TES2LIB     76/09/16. 08.09.26    NOS 1.1   FTN     4.6433  666X  T        OPT=1
          D           REL       TES2LIB     76/09/16. 08.09.26    NOS 1.1   FTN     4.6433  666X  T        OPT=1
ADDED     LIBRARY     OPLD      *****       76/09/16.
          **EOF**               TES2LIB
```

The LIBGEN statement generates a user library, using this new program library (written by LIBEDIT on the file NEW) as the source file. The user library is written on the file LIBLOAD and has the name LOADLIB.

The CATALOG of the user library shows the following content.

```
       CATALOG OF LIBLOAD        FILE     1                                76/09/16. 08.10.31.      PAGE        1
REC    NAME       TYPE         LENGTH     CKSUM      DATE        COMMENTS

 1     LOADLIB    ULIB            11      4752   76/09/16.
 2     B          REL             30      5411   76/09/16. 08.08.58    NOS 1.1   FTN     4.6433  666X  T          OPT=1
 3     BONE       REL             30      0677   76/09/16. 08.09.26    NOS 1.1   FTN     4.6433  666X  T          OPT=1
 4     NEWC       REL             30      1377   76/09/16. 08.09.26    NOS 1.1   FTN     4.6433  666X  T          OPT=1
 5     D          REL             30      6030   76/09/16. 08.09.26    NCS 1.1   FTN     4.6433  666X  T          OPT=1
 6     LOADLIB    OPLD            13      4335   76/09/16.

 7     * EOF *         SUM =     164
```

Example 5:

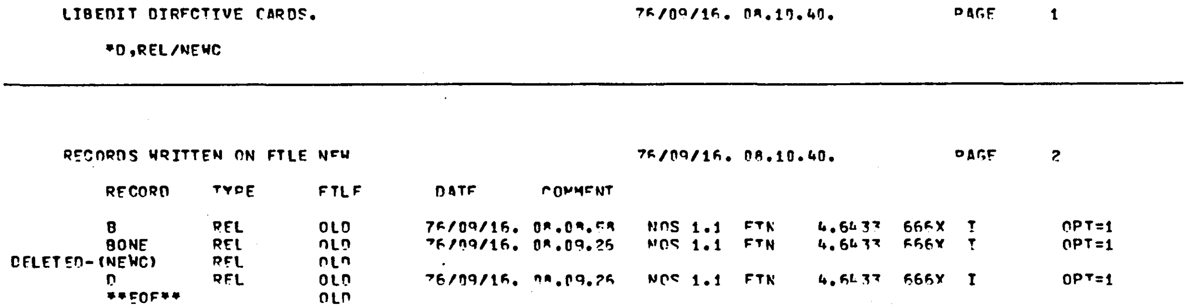
This job illustrates a technique of deleting records from a user library, which is an alternative to the method shown in example 4. This alternate method uses GTR to collect the relocatable records (REL) from the user library and makes the desired deletions from this file using LIBEDIT. Then, LIBGEN generates a user library, using this modified copy as the source.

```
LIBTES4.
USER, EFD25.
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.

The GTR statement reads the relocatable records from LIBLOAD and writes them on the file OLD. (This control statement terminates after OLD; the REL/* is a directive specifying all relocatable records.)

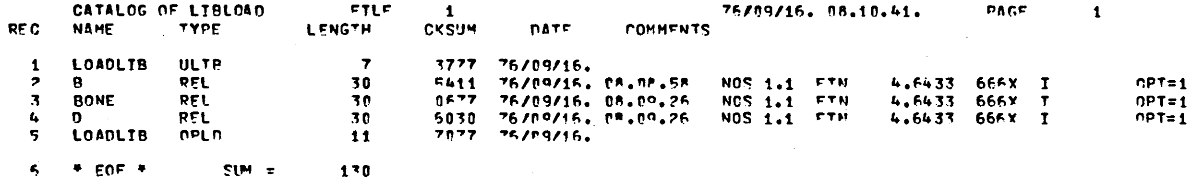
LIBEDIT references the program library OLD and the directive record, deletes NEWC, and writes this modified file on the default NEW. The following is a listing of NEW.



```
      LIBEDIT DIRECTIVE CARDS.                                 76/09/16. 08.10.40.          PAGE     1

          *D,REL/NEWC

--------------------------------------------------------------------------------------------------------


      RECORDS WRITTEN ON FILE NEW                              76/09/16. 08.10.40.          PAGE     2

          RECORD     TYPE       FILE       DATE      COMMENT

          B          REL        OLD      76/09/16. 08.08.58   NOS 1.1  FTN     4.6433  666X  I        OPT=1
          BONE       REL        OLD      76/09/16. 08.09.26   NOS 1.1  FTN     4.6433  666X  I        OPT=1
   DELETED-(NEWC)    REL        OLD          .
          D          REL        OLD      76/09/16. 08.09.26   NOS 1.1  FTN     4.6433  666X  I        OPT=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 to show the following contents.



```
        CATALOG OF LIBLOAD        FILE     1                              76/09/16. 08.10.41.       PAGE       1
   REC  NAME      TYPE          LENGTH    CKSUM      DATE      COMMENTS

    1   LOADLIB   ULIB             7       3777   76/09/16.
    2   B         REL             30       5411   76/09/16. 08.08.58   NOS 1.1  FTN     4.6433  666X  I          OPT=1
    3   BONE      REL             30       0677   76/09/16. 08.09.26   NOS 1.1  FTN     4.6433  666X  I          OPT=1
    4   D         REL             30       5030   76/09/16. 08.09.26   NOS 1.1  FTN     4.6433  666X  I          OPT=1
    5   LOADLIB   OPLD            11       7077   76/09/16.

    6   * EOF *         SUM =    130
```

Example 6:

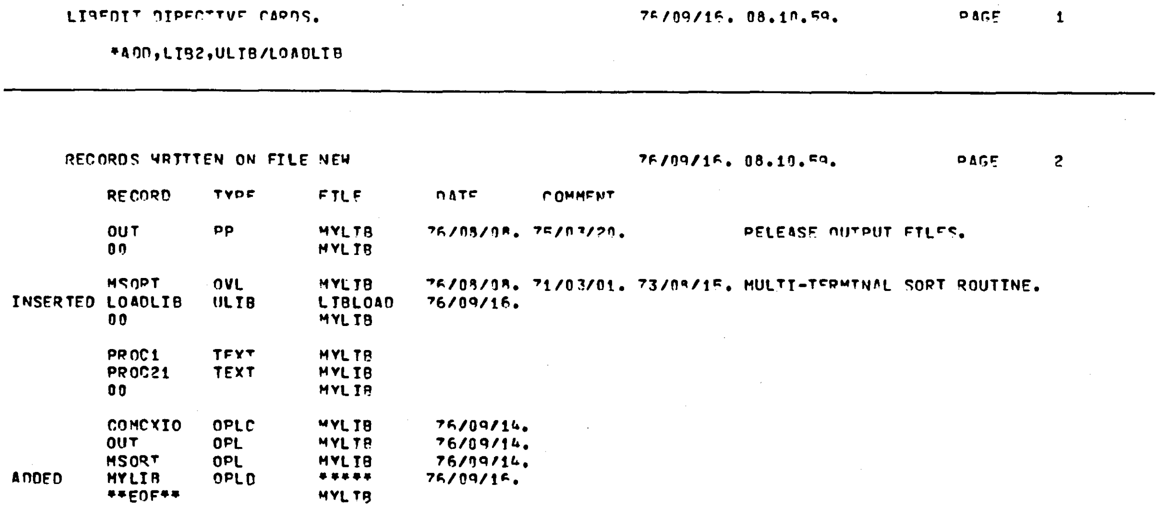
This job uses LIBEDIT to add a user library to a program library. The user library is the version of LIBLOAD generated in example 4. The program library is an existing file, MYLIB. Since LIBEDIT manipulates ULIB type programs as a single unit, the entire user library is added to the program library. LIBEDIT cannot access individual elements within a user library.

```
LIBTES6.
USER, EFD25.
CHARGE, 16, 13N122.
ATTACH, MYLIB.
ATTACH, LIBLOAD.
DEFINE(NEW=NEWLIB)
LIBEDIT, P=MYLIB, B=LIBLOAD.
CATALOG, NEW, R.
CATALOG, NEW, R, U.
/EOR
*ADD, LIB2, ULIB/LOADLIB
/EOF
```

MYLIB and LIBLOAD are attached to the job's control point.

A direct access file NEWLIB is defined. It is referenced as NEW so it can serve as the default file on which the new program library will be written.

LIBEDIT reads the replacement file LIBLOAD and the input directive. The directive specifies that the addition will be before the zero-length record that terminates the second library on the old program library. The addition will be the user library LOADLIB. The new program library is written on NEW with an updated file directory added at the end. The following listing of NEW is written to OUTPUT.



```
     LIBEDIT DIRECTIVE CARDS.                                 76/09/16. 08.10.59.          PAGE     1

         *ADD,LIB2,ULIB/LOADLIB
──────────────────────────────────────────────────────────────────────────────────────────────────


     RECORDS WRITTEN ON FILE NEW                              76/09/16. 08.10.59.          PAGE     2

         RECORD    TYPE     FILE      DATE      COMMENT

         OUT       PP       MYLIB     76/08/08. 75/03/20.           RELEASE OUTPUT FILES.
         00                 MYLIB

         MSORT     OVL      MYLIB     76/08/08. 71/03/01. 73/08/15. MULTI-TERMINAL SORT ROUTINE.
INSERTED LOADLIB   ULIB     LIBLOAD   76/09/16.
         00                 MYLIB

         PROC1     TEXT     MYLIB
         PROC21    TEXT     MYLIB
         00                 MYLIB

         COMCXIO   OPLC     MYLIB      76/09/14.
         OUT       OPL      MYLIB      76/09/14.
         MSORT     OPL      MYLIB      76/09/14.
ADDED    MYLIB     OPLD     *****     76/09/16.
         **EOF**            MYLIB
```

The first CATALOG shows the following content of the new program library.

```
        CATALOG OF NEW           FILE     1                            76/09/16. 08.11.00.      PAGE       1
  REC   NAME      TYPE          LENGTH    CKSUM      DATE      COMMENTS

    1   OUT       PP (1100)        255    6220  76/08/08. 75/03/20.           RELEASE OUTPUT FILES.
    2   (00)           SUM =       255

    3   MSORT     OVL 00,00        252    0720  76/08/09. 71/03/01. 73/08/15. MULTI-TERMINAL SORT ROUTINE.
    4   LOADLIB   ULIB               7    3777  76/09/16.
    9   (00)           SUM =       412

   10   PROC1     TEXT               4    7015
   11   PROC21    TEXT              44    1652
   12   (00)           SUM =        50

   13   COMCXIO   OPLC (64)        513    0003   76/09/14.
   14   OUT       OPL  (64)       1615    1411   76/09/14.
   15   MSORT     OPL  (64)       2042    3074   76/09/14.
   16   MYLIB     OPLD              23    4474  76/09/16.

   17   * EOF *        SUM =      5354
```

The entire user library is added to the program library.

The second CATALOG specifies the U parameter, which includes the records of the user library in the following listing. These records begin a second page in the printout.

```
        CATALOG OF NEW           FILE     1                            76/09/16. 08.11.01.      PAGE       1
  REC   NAME      TYPE          LENGTH    CKSUM      DATE      COMMENTS

    1   OUT       PP (1100)        255    6220  76/08/08. 75/03/20.           RELEASE OUTPUT FILES.
    2   (00)           SUM =       255

    3   MSORT     OVL 00,00        252    0720  76/08/09. 71/03/01. 73/08/15. MULTI-TERMINAL SORT ROUTINE.
```

```
        CATALOG OF NEW           FILE     1                            76/09/16. 08.11.01.      PAGE       2
  REC   NAME      TYPE          LENGTH    CKSUM      DATE      COMMENTS

    4   LOADLIB   ULIB               7    3777  76/09/16.
    5   B         REL               30    5411  76/09/16. 08.08.58   NOS 1.1  FTN     4.6433  666X  I          OPT=1
    6   BONE      REL               30    0677  76/09/16. 08.09.26   NOS 1.1  FTN     4.6433  666X  I          OPT=1
    7   D         REL               30    6030  76/09/16. 08.09.26   NOS 1.1  FTN     4.6433  666X  I          OPT=1
    8   LOADLIB   OPLD              11    7077  76/09/16.
    9   (00)           SUM =       412

   10   PROC1     TEXT               4    7015
   11   PROC21    TEXT              44    1652
   12   (00)           SUM =        50

   13   COMCXIO   OPLC (64)        513    0003   76/09/14.
   14   OUT       OPL  (64)       1615    1411   76/09/14.
   15   MSORT     OPL  (64)       2042    3074   76/09/14.
   16   MYLIB     OPLD              23    4474  76/09/16.

   17   * EOF *        SUM =      5354
```

JOB OUTPUT INFORMATION 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(CM50000,T10)
USER(JEANCOM,PASSWOR,SYS172)
FTN.
-EOR-
      PROGRAM CONVER(INPUT,OUTPUT)
C         THIS PROGRAM CONVERTS OCTAL TO DECIMAL
C         THE SECOND VALUE PRINTED IS 10 OCTAL TIMES THE FIRST
C         TERMINATE BY TYPING ZERO
   2 CONTINUE
      READ 1,J
   1 FORMAT(O8)
      K=J*10B
      PRINT 6,J,K
   6 FORMAT(5X,I10,5X,I10)
      IF(J.EQ.0)3,2
   3 CONTINUE
      STOP
      END
-EOI-
```

```
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   JJJJJ         CCCCCCCCCC
```

The first four characters of the banner job name are generated from the user index associated with the user number. These four characters are unique to each user and remain the same for subsequent jobs run under the same user number. The last three characters are the job sequence number assigned by the system at the time of processing.

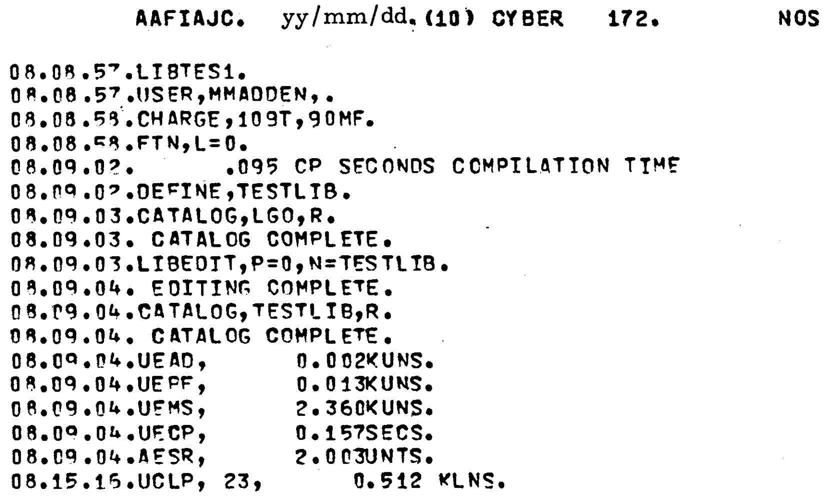
```
yy/mm/dd.  hh.mm.ss.
```

This line specifies the current date (year/month/day.) and the time (hours. minutes. seconds.) when job printing was initiated.

```
   PROGRAM CONVER     73/74    OPT=1                          FTN 4.4+U401       yy/mm/dd.  hh.mm.ss.     PAGE     1

1                  PROGRAM CONVER(INPUT,OUTPUT)
           C          THIS PROGRAM CONVERTS OCTAL TO DECIMAL
           C          THE SECOND VALUE PRINTED IS 10 OCTAL TIMES THE FIRST
           C          TERMINATE BY TYPING ZERO
5                2 CONTINUE
                   READ 1,J
                 1 FORMAT(O8)
                   K=J*10B
                   PRINT 6,J,K
10               6 FORMAT(5X,I10,5X,I10)
                   IF(J.EQ.0)3,2
                 3 CONTINUE
                   STOP
                   END
```

The job calls the FORTRAN Extended compiler which compiles the program, CONVER, contained in the program record. The symbolic reference map for program CONVER is printed below.

```
      SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
 4107  CONVER

VARIABLES      SN  TYPE            RELOCATION
 4137  J          INTEGER                           4140  K         INTEGER

FILE NAMES        MODE
    0  INPUT      FMT           2041  OUTPUT    FMT

STATEMENT LABELS
 4124  1        FMT                4110  2                               0  3           INACTIVE
 4133  6        FMT

STATISTICS
  PROGRAM LENGTH              36B      30
  BUFFER LENGTH             4103B    2115
```



```
          AAFIAJC.   yy/mm/dd.(10) CYBER    172.          NOS

08.08.57.LIBTES1.
08.08.57.USER,MMADDEN,.
08.08.58.CHARGE,109T,90MF.
08.08.58.FTN,L=0.
08.09.02.       .095 CP SECONDS COMPILATION TIME
08.09.02.DEFINE,TESTLIB.
08.09.03.CATALOG,LGO,R.
08.09.03. CATALOG COMPLETE.
08.09.03.LIBEDIT,P=0,N=TESTLIB.
08.09.04. EDITING COMPLETE.
08.09.04.CATALOG,TESTLIB,R.
08.09.04. CATALOG COMPLETE.
08.09.04.UEAD,      0.002KUNS.
08.09.04.UEPF,      0.013KUNS.
08.09.04.UEMS,      2.360KUNS.
08.09.04.UECP,      0.157SECS.
08.09.04.AESR,      2.003UNTS.
08.15.16.UCLP, 23,      0.512 KLNS.
```

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, 0.013 kilounit of permanent file activity, 2.360 kilounits of mass storage activity, 0.157 seconds of central processor time, and 2.003 SRU. The job produced 0.512 kiloline (512) of printable output. Depending on the resources used, additional information may be included in the dayfile. Refer to Job Completion, section 3 for the formats of these messages.

PERMANENT FILE DEVICE STATISTICS

The system allocates space for permanent mass storage files in units called reservation blocks. The size of a reservation block depends upon the type of file and/or the type of device on which the file is to reside. For indirect access files, the reservation block size is always one PRU (64 CM words), regardless of the device residence. For direct access files, the reservation block size is a multiple of PRUs and varies according to the device type, as shown in the following table.

Device Type	Device	PRUs/ Block	CM Words	Characters	Maximum No. of Blocks
DE	Extended Core Storage	16	1024	10,240	121 for 125K 243 for 250K
DI	844-21 Disk Storage Subsystem (1≤n≤8)	n*107	n*6,848	n*68,480	1616
DJ	844-41/44 Disk Storage Subsystem (1≤n≤8)	n*227	n*14,528	n*145,280	1640
DP	Distributive Data Path to ECS	16	1024	10,240	121 for 125K 243 for 250K
MD	841-n Multiple Disk Drive (1≤n≤8)	n*32	n*2048	n*20,480	1600

In this table, n indicates the unit count for multiunit devices.

In general, the largest permanent file the user can create is a direct access file that resides on a nonmaster device within his family of permanent file devices. Such files are restricted in size by the limitations of the device itself and the DS validation parameter which limits the size of direct access files. If no DS restriction is imposed, the maximum file size equals the maximum number of reservation blocks that can be allocated for the device.

All other permanent files reside either on the user's master device or on an auxiliary device. Their maximum size is restricted to the device limit minus any space allocated for catalog information and other files. In addition, an installation can use the FS validation parameter to limit the size of indirect access files.

CARD FORMAT AND CONVERSION PROBLEMS F

Data within the system is stored in binary or coded records. Binary records are variable in length and consist of central memory images. Coded records consist of lines of display-coded characters. Binary and coded data can enter the system in several different formats. Some of these formats can enter the system directly; others must be converted into a system-recognizable format. In either case, there are several formats in which the data can reside in the system. Accordingly, the processing program must take into account the specific format of data it accesses.

This appendix describes the formats for punched cards and the format for printed data. It also describes the conversion performed by the system on data transferred between the system and peripheral devices and the method by which time-sharing terminal data is converted in the system.

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 bits of zero, 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 would appear in memory as follows:

```
59            47            35            23            11           0
|   00 00    |   00 00     |   00 00     |   00 00     |   00 01     |

     :  :         :  :          :  :          :  :          :  A

|   00 00    |   00 00     |   00 00     |   00 00     |   01 01     |

     :  :         :  :          :  :          :  :          A  A

|   00 00    |   00 00     |   00 00     |   00 00     |   00 00     |

                          end-of-line
```

However, if the characters were copied with the COPYSBF utility, the following would appear.

```
59         47         35         23         11          0
| 55 00    | 00 00    | 00 00    | 00 00    | 00 00    |

                    end-of-line

| 01 00    | 00 00    | 00 00    | 00 00    | 00 01    |

   A  :       :  :       :  :       :  :       :  A

| 01 00    | 00 00    | 00 00    | 00 00    | 00 00    |

   A                end-of-line
```

NOTE

If a colon is the last character of a line, the system appends a blank character to preserve the colon and then appends an end-of-line (two blanks may be added to ensure an even number of characters).

FORMATS FOR CARDS READ

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 the following section.

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 spaces are deleted.

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 all subsequent coded cards are converted in O26 mode. A 29 indicates subsequent

†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.)

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 punched 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.

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

FORMATS FOR CARDS PUNCHED

Punched cards can be in three formats.

- Coded (punch Hollerith)
- 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.

PRINTED DATA

All printed data is in coded format. The line printer driver extracts data until an end-of-line mark occurs or until 14 central memory words are retrieved. The end-of-line is denoted by a zero byte as the last byte of a central memory word. The print line consists of a maximum of 136 characters. If an end-of-line mark does not appear after 136 characters, the last four characters of that group are lost. The driver converts the extracted data from display code to internal BCD code (refer to appendix A for the character set equivalences) and forwards the data to the line printer controller.

The driver interprets the first character in a line as the carriage control character (refer to appendix A) and that character is not printed. In most cases, the proper carriage control is issued while the remainder of the line is printed. However, when Q, R, S, or T is specified, no printing takes place for that line. The Q, R, S, and T format controls remain in effect until changed, and all other carriage control options must be supplied for each line they control. Line spacing is normally done in the auto eject mode; that is, creases in the paper are skipped by the line printer controller's automatic line spacing mechanism if the paper is loaded properly. Auto eject mode must be deselected if the user wants to employ 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/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/skipped on a page, the line limit is still decremented by the default number of lines.

TERMINAL CHARACTER CONVERSION

Normal input mode from a terminal consists of a 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 terminals which NOS supports can be grouped into ASCII terminals and correspondence code terminals. The manner in which the characters entered from a terminal are interpreted by the system depends on whether the user specifies that the characters belong to the full character set. For example, if the user enters the following characters to be mapped into the full ASCII set

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 a NORMAL command is issued, the characters are mapped into the subset of the ASCII character set; 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

In ASCII mode, †† all 128 characters from an ASCII or correspondence code terminal are recognized. These characters, in addition to the first 64, are processed as 12-bit characters with an escape code convention as shown previously. Table 1-A-1 lists the character set equivalences. The programs that process data must recognize that data is in ASCII mode rather than normal display code and process it accordingly.

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. All information is transmitted in even parity unless the user specifies odd parity.

For a more detailed description of terminal operation, refer to the Time-Sharing User's Reference Manual.

Data can also be transmitted to or from a terminal through a paper tape reader. The paper tape character mode is always ASCII.

† Partial words are zero-filled; partial bytes are blank-filled.

†† Refer to the Time-Sharing User's Reference Manual for descriptions of the ASCII and NORMAL commands.

TAPE LABELS G

The operating system accepts ANSI standard and nonstandard labeled tapes. Labels which do not conform to ANSI standards in format and/or content are defined as nonstandard.

ANSI labels perform two functions. They provide information that uniquely identifies a file and the reel on which it resides, and they mark the beginning and end 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 in length and are recorded at the same density as the data on the tape. The first three 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 At	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-file	Required
HDR	2-9	File header label	Beginning-of-file	Optional
UHL	†	User header label	Beginning-of-file	Optional
EOF	1	End-of-file label	End-of-file	Required
EOF	2-9	End-of-file label	End-of-file	Optional
UTL	†	User trailer label	End-of-file	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 will be written as specified. Note that 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.

<table>
<tr><td>VOL</td><td>1</td><td>volume serial number</td></tr>
<tr><td>va</td><td colspan="2">reserved</td></tr>
<tr><td colspan="3">reserved</td></tr>
<tr><td colspan="2">reserved</td><td>owner identification</td></tr>
<tr><td colspan="3">owner identification (oid)</td></tr>
<tr><td>oid</td><td colspan="2">reserved</td></tr>
<tr><td colspan="3">reserved</td></tr>
<tr><td colspan="2">reserved</td><td>lsl</td></tr>
</table>

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.

<table>
<tr><td colspan="2">HDR</td><td>1</td><td colspan="4">file identifier (fi)</td></tr>
<tr><td colspan="7">file identifier (fi)</td></tr>
<tr><td>fi</td><td colspan="4">set identification</td><td colspan="2">file section number (secno)</td></tr>
<tr><td>secno</td><td colspan="3">file sequence number</td><td colspan="2">generation number</td><td>gvn</td></tr>
<tr><td>gvn</td><td colspan="4">creation date</td><td colspan="2">expiration date</td></tr>
<tr><td colspan="2">expiration date</td><td>fa</td><td colspan="4">block count</td></tr>
<tr><td colspan="7">system code</td></tr>
<tr><td colspan="2">system code</td><td colspan="5">reserved</td></tr>
</table>

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	File identification (fileid) parameter on the LABEL control statement	Blank	Checked if specified
22-27	Set identification	6	Set identification as specified by the setid parameter on the LABEL control statement. This value must be the same for all files of a multifile set.	Blank	Checked if specified
28-31	File section number (secno)	4	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	Position of a file within a file set, as specified by the seqno parameter of the LABEL statement. 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 will contain 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	Generation number of a file, as specified by 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.

EOF	1	file identifier (fi)	
file identifier (fi)			
fi	set identification	file section number (secno)	
secno	file sequence number	generation number	gvn
gvn	creation date	expiration date	
expiration date	fa	block count	
system code			
system code	reserved		

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.

<table>
<tr><td colspan="2">EOV</td><td>1</td><td colspan="4">file identifier (fi)</td></tr>
<tr><td colspan="7">file identifier (fi)</td></tr>
<tr><td>fi</td><td colspan="4">set identification</td><td colspan="2">file section number (secno)</td></tr>
<tr><td>secno</td><td colspan="3">file sequence number</td><td colspan="2">generation number</td><td>gvn</td></tr>
<tr><td>gvn</td><td colspan="4">creation date</td><td colspan="2">expiration date</td></tr>
<tr><td colspan="2">expiration date</td><td>fa</td><td colspan="4">block count</td></tr>
<tr><td colspan="7">system code</td></tr>
<tr><td colspan="2">system code</td><td colspan="5">reserved</td></tr>
</table>

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

These labels define four possible file configurations.

- A single file on a single volume
- A single file on two or more volumes
- Two or more files on a single volume
- Two or more files on two or more volumes

Figures 1-G-1 through 1-G-7 illustrate the use of ANSI labels in these configurations.

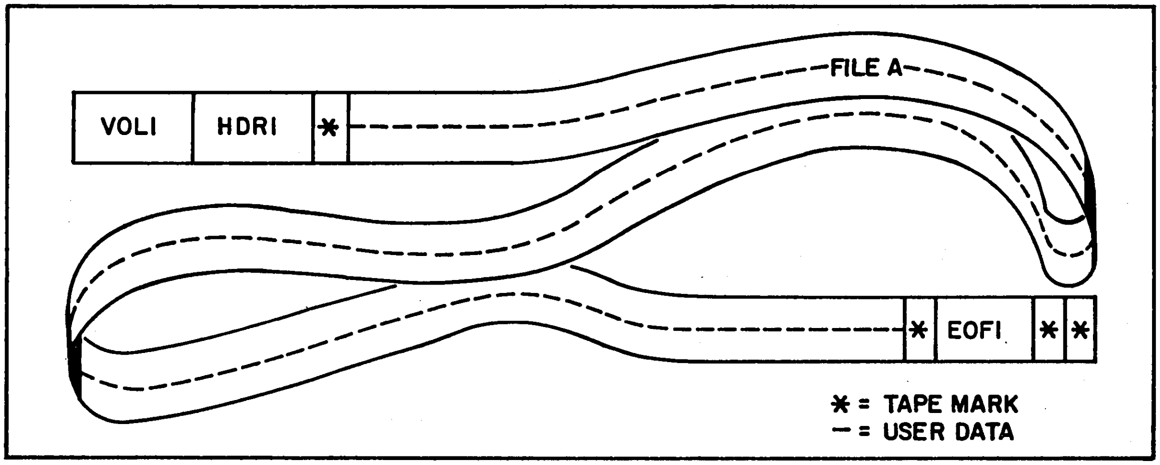


Figure 1-G-1. ANSI Labels: Single File, Single Volume

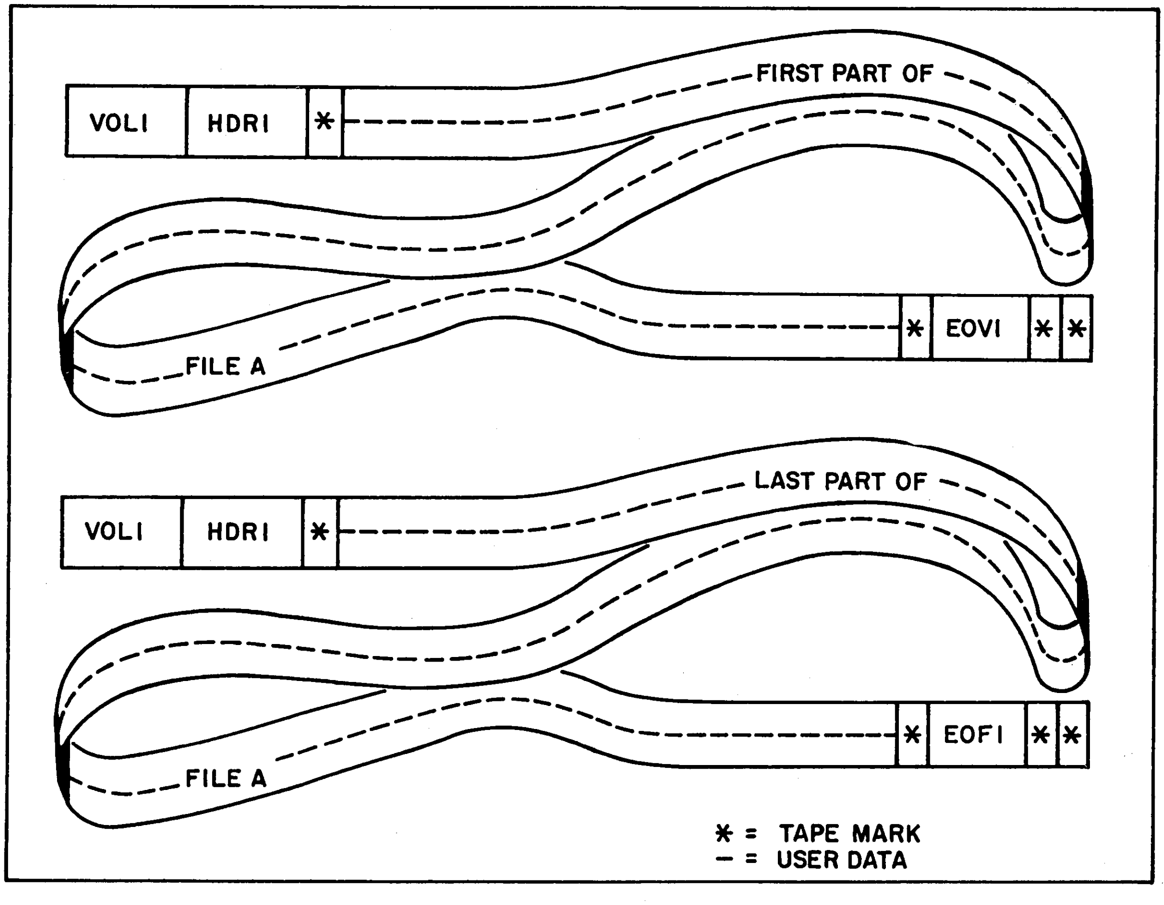


Figure 1-G-2. ANSI Labels: Single File, Multivolume

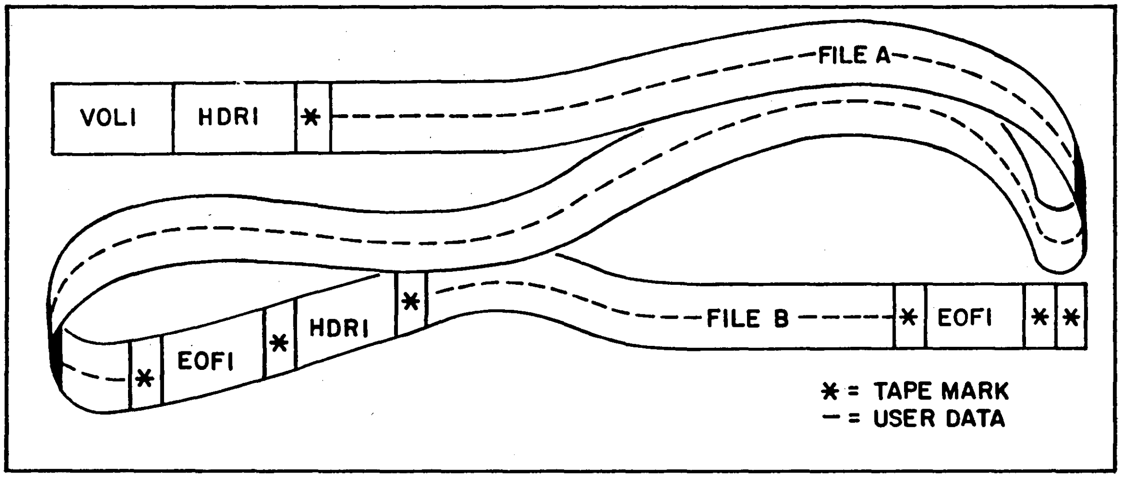


Figure 1-G-3 ANSI Labels: Multifile, Single Volume

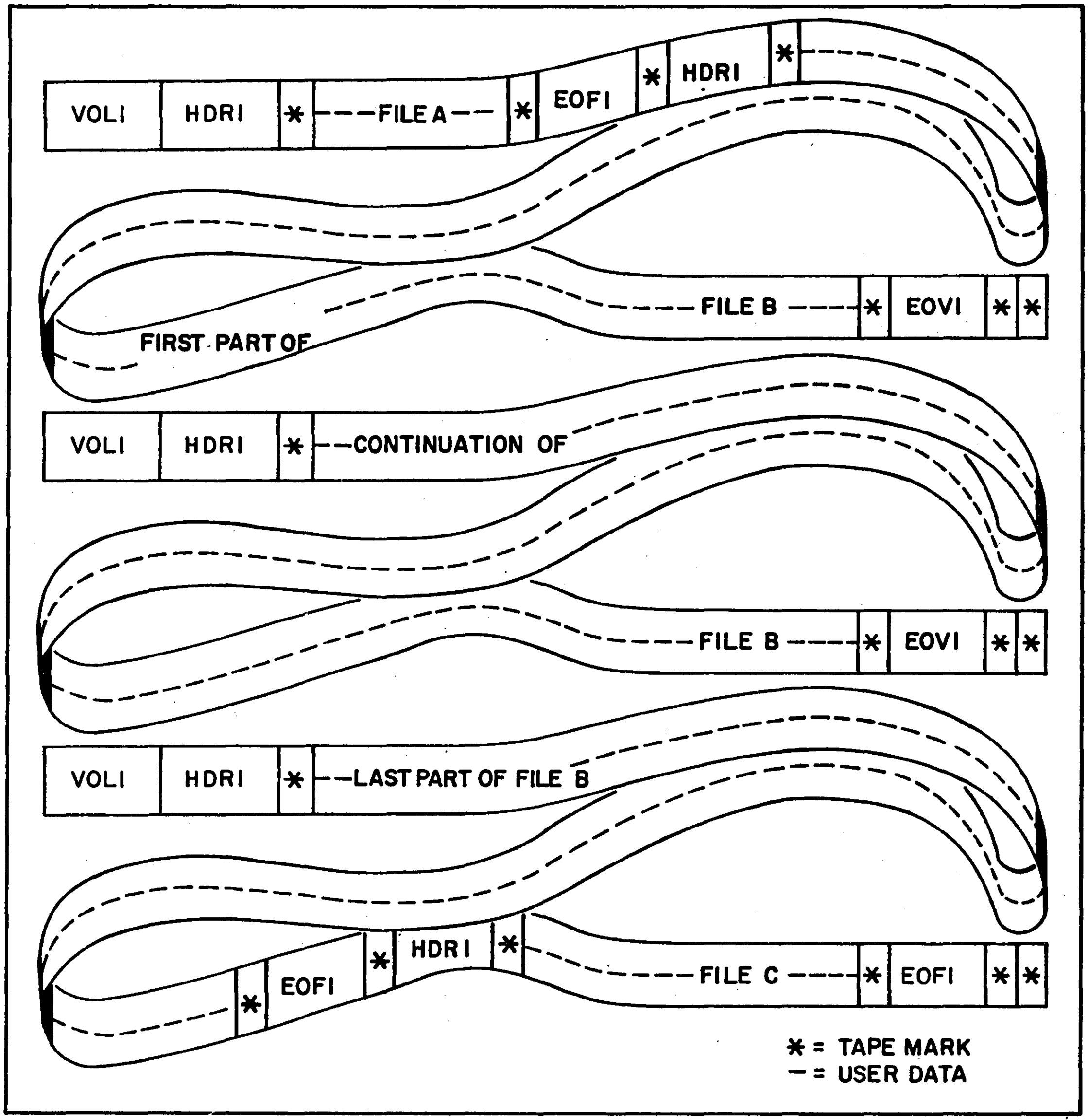


Figure 1-G-4. ANSI Labels: Multifile, Multivolume

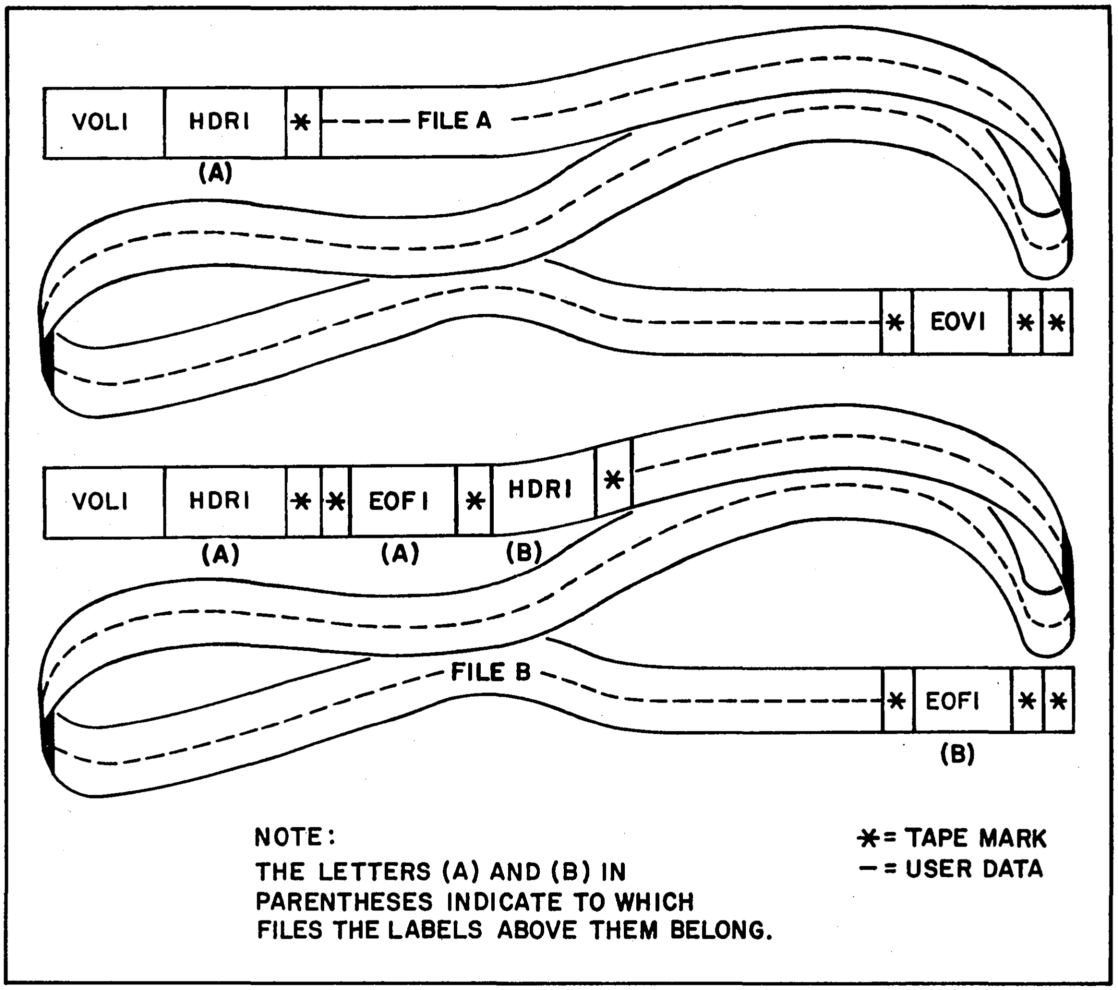


Figure 1-G-5. ANSI Labels: End-of-File, End-of-Volume Coincidence

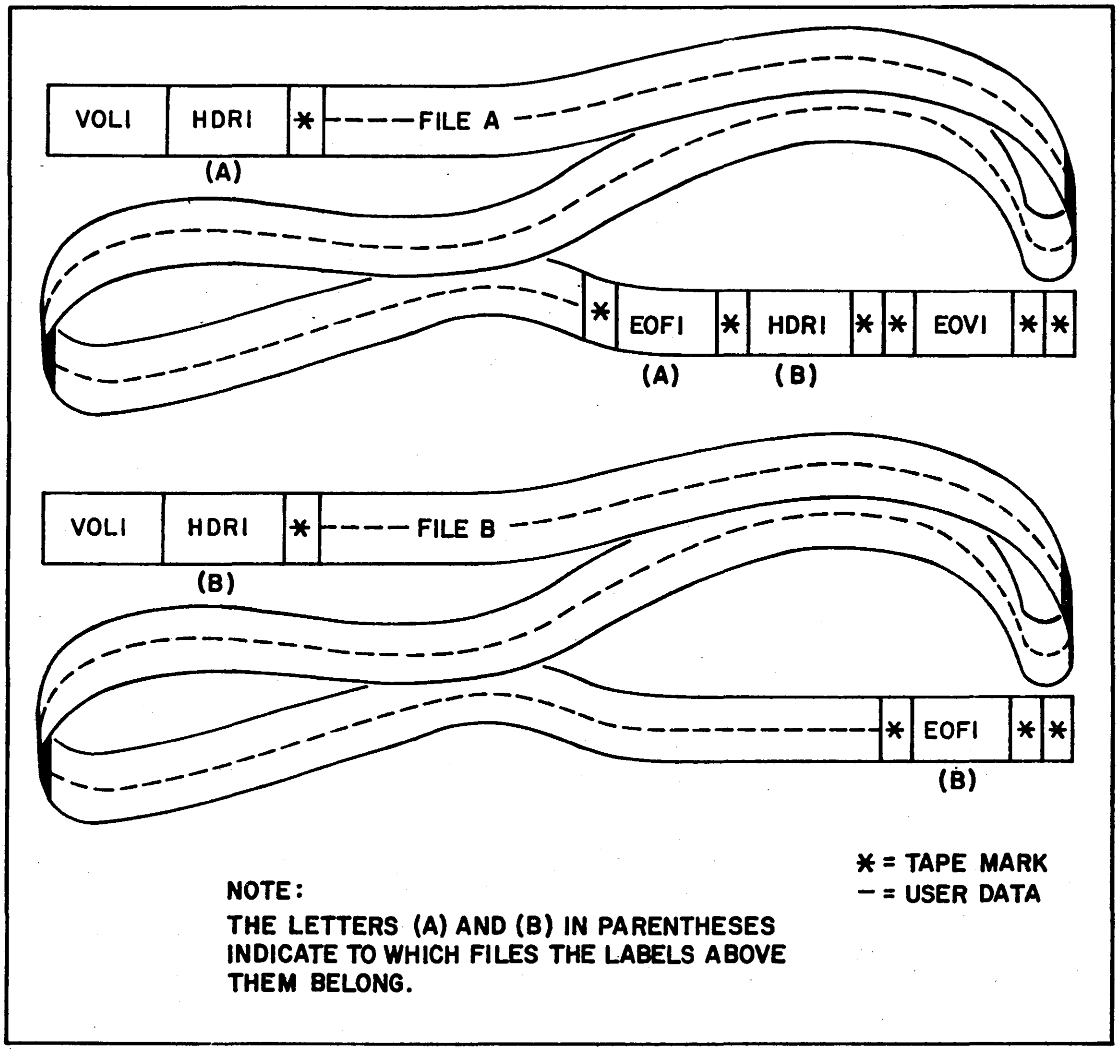


Figure 1-G-6. ANSI Labels: End-of-File, End-of-Volume Coincidence

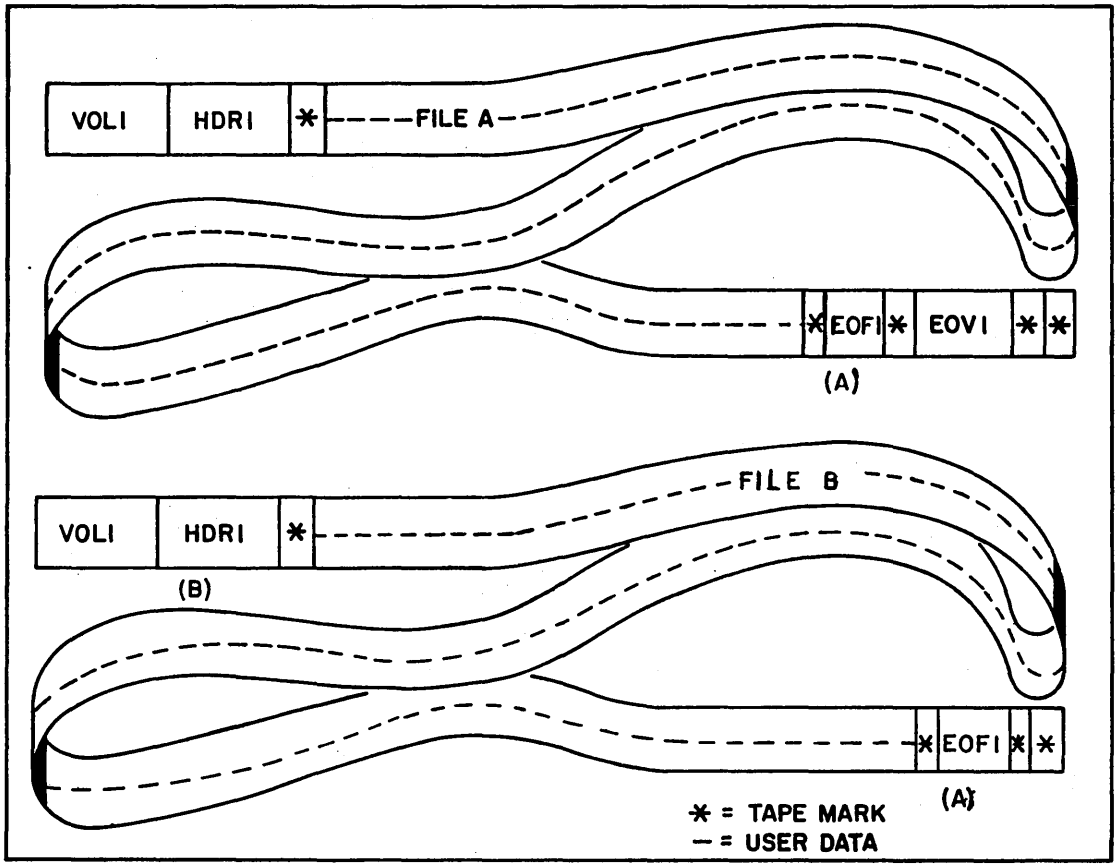


Figure 1-G-7. ANSI Labels: End-of-File, End-of-Volume Coincidence

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.

EOV2-9 — ADDITIONAL END-OF-VOLUME LABELS

EOV2-9 labels may immediately follow EOV1. Their format is:

Character Position	Field Name	Length (in characters)	Contents	Default Written
1-3	Label identifier	3	EOV	EOV
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, 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, etc., 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.

GLOSSARY H

BLOCK

The information between interrecord gaps on an NOS tape format. This term is not defined for operating system mass storage devices. In CDC CYBER Record Manager, there are four block types for sequential files. Blocking is the grouping of user records for efficiency in transfer between memory and storage devices.

BOI

Beginning-of-information.

CDC CYBER RECORD MANAGER

A software product supported under NOS that allows a variety of record types, blocking types, and file organizations to be created and read. The execution time input/output of COBOL 4, COBOL 5, FORTRAN Extended 4, Sort/Merge 4, ALGOL 4, BASIC, and the DMS-170 products is implemented through CDC CYBER Record Manager. The system input/output of NOS is not implemented through CDC CYBER Record Manager. All CDC CYBER Record Manager file processing requests ultimately pass through the operating system input/output routines. COMPASS programs can use either CDC CYBER Record Manager or NOS input/output (CIO).

CIO

Combined input/output performs input/output for NOS. The data formats (physical and logical) do not necessarily match the data formats used by CDC CYBER Record Manager.

CONTROL STATEMENT RECORD

The first, and possibly only, record on an INPUT file or a deferred batch job file consisting of statement images that start with a job statement and end with the first EOR, EOF, or EOI.

DIRECT ACCESS FILE

A permanent file that can be attached to the 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).

EOF

End-of-file is a boundary within a sequential file, but not necessarily the end of a file that can be referenced by name. The actual end of a named file is defined by EOI. On a PRU device, a zero length PRU with level designator of 17 indicates EOF. On tapes other than SI, I, or X format, EOF is represented by a tape mark (refer to section 10).

In CDC CYBER Record Manager, a zero length PRU with a level designator of 17 and a tape mark on a tape in S or L format is a partition boundary. For W type records, the partition boundary is marked by a W control word with the end of partition flag set. CDC CYBER Record Manager divides files into partitions; therefore, an NOS multifile file is a multipartition file when discussing PRU devices.

EOI

End-of-information.

EOR

End-of-record is the terminator of a logical record. On a PRU device, a short PRU or a zero length PRU with a level designator of 0 indicates EOR. On tapes that are not PRU devices, an interrecord gap indicates EOR. Only CDC CYBER Record Manager S type terminator is equivalent to the NOS EOR boundary.

EMPTY RECORD

Refer to zero length PRU.

FET

The file environment table defines the current status and properties of a file that is being used by a job. CDC CYBER Record Manager uses an FIT to describe its files and interfaces to CIO through the FET.

FILE

Set of information that begins at BOI and ends at EOI and that is referred to by a logical file name. This is the only definition of a file in CDC CYBER Record Manager and the languages that use CDC CYBER Record Manager. In NOS, a file is also defined as that portion of a file terminated by EOF; thus, a multifile file can exist. Generally, when an NOS control statement has a parameter that is a file name, that parameter refers to the BOI and EOI definition. When an NOS control statement has a parameter that specifies the number of files, that parameter uses the EOF definition.

FILE TYPES

In CDC CYBER Record Manager, one of five file organizations. In NOS, file types has several meanings depending upon context (refer to section 2 for a description of NOS file types).

FIT

File information table is required by CDC CYBER Record Manager for each file to be accessed. Fields in the table describe such items as file structure and record type. All CDC CYBER Record Manager input/output is based on the content of the table. CDC CYBER Record Manager provides the interface between the FIT and FET. NOS uses the FET for input/output to a device.

INDIRECT ACCESS FILE

A permanent file that is accessed only by making a working copy of the file (GET or OLD control statements). It is created or altered by saving or substituting the contents of an existing working file (REPLACE or SAVE control statements).

INTERRECORD GAP

Physical spaces between data blocks on tape.

LEVEL DESIGNATOR

The level designator is an octal number in the terminating marker of a PRU, ranging from 00 to 17_8. A level 17 in an empty PRU designates an EOF in NOS and an end of partition in CDC CYBER Record Manager. A level 0 in a short PRU designates an EOR in NOS. A level 1 in a short PRU in NOS designates an EOR and that the record came from an interactive terminal. A level 16 in a short PRU in NOS designates an EOR on a checkpoint file.

LINE

Refer to zero byte terminator.

LOCAL FILE

A file that is currently associated with a job.

LOGICAL RECORD

A logical record on mass storage is terminated by an EOR; on tape, it is terminated by the conditions described in section 10 for individual tape formats. Often, a logical record contains more than one CDC CYBER Record Manager record. Since CDC CYBER Record Manager defines a line as a logical record, an NOS logical record may contain several record manager logical records.

PARTITION

A CDC CYBER Record Manager file with sequential and word addressable organizations. It represents a file division that can contain records and sections. A file may have one or more partitions.

The physical representation of a partition on an S or L tape is a tape mark. On a PRU device, a file with record type other than W has partitions indicated by a zero length record with a level designator of 17. For W type records, a partition is not equivalent to any designator recognized by NOS.

PRIMARY FILE

Any working file created with the OLD, PRIMARY, or NEW control statement.

PROCEDURE FILE

A file containing control language and control statements.

PRU

Physical record unit. The amount of information transmitted by a single physical operation of a specified device. A PRU for mass storage devices is 64 central memory words in length. A PRU for a binary tape in SI, I, or X format is 512 central memory words long, and a PRU for a coded tape in SI, I, or X format is 128 central memory words long. It may not be full of user data (short PRU) or may contain no user data (empty or zero length PRU). The physical length of the PRU is as previously defined.

PRU DEVICE

Any mass storage device or tape in SI, I, or X format, where these records are written in PRUs.

RANDOM FILE

A file with an address associated with each record such that a particular record in the file can be accessed by address. To be accessed randomly, a file must reside on mass storage. NOS recognizes a file as being random only when the random bit is set in the FET. CDC CYBER Record Manager recognizes four types of random access files: word addressable, indexed sequential, direct access, and actual key organizations. All CDC CYBER Record Manager organizations are sequential files when processed by NOS.

RECORD

A unit of information, which is interchangeable with logical records in NOS.

In CDC CYBER Record Manager and its language processors, a unit of information produced by a single write request. In FORTRAN Extended, a formatted write produces zero byte terminated records, and an unformatted write produces W type records. An operating system record is not the same as a CDC CYBER Record Manager type record unless the CDC CYBER Record Manager record type is declared to be S.

RECORD SEPARATOR

In NOS, another name for an EOR.

RECORD TYPE

May have one of several meanings, depending upon its context. In CDC CYBER Record Manager, there are seven record types defined by the RT field in the FIT.

SEQUENCE NUMBERS

Line numbers at the beginning of each line of a file. If a file uses sequence numbers, zero byte terminated records are implied.

SEQUENTIAL FILE

A file in which records are accessed in the logical order in which they occur. Any file can be accessed sequentially. Sequential files must be accessed sequentially because no key or address is associated with each record in the files. All CDC CYBER Record Manager files are considered sequential files by NOS.

SHORT PRU

A PRU that does not contain the maximum number of character data allowed for that device.

W TYPE RECORD

A CDC CYBER Record Manager record type in which user data is preceded by a system-supplied control word. FORTRAN Extended unformatted writes and Sort/Merge use W type records as default record types. EOF and partition boundaries are not equivalent on files with this type of record.

WORKING FILE

A file that is currently associated with a job and is temporary in nature. That is, all working files cease to exist once they are returned to the system (either specifically or at job termination).

ZERO BYTE TERMINATOR

The 12 bits of zero in the low order position of a central memory word are used to terminate a line of coded information to be output to a line printer or to represent cards input through a card reader. Files with names INPUT and OUTPUT have such terminators while in storage. Any file to be displayed at a terminal must also have such terminators for each line to be displayed correctly. A record with such a terminator in CDC CYBER Record Manager is a zero-byte record (Z type record).

The COPYSBF, COPYCR, LO72, LIST80, RESEQ, ROUTE, and SUBMIT control statements require files whose lines are zero-byte records. A record (marked by EOR) in NOS may contain one or several zero-byte records.

In display code, two colons create 12 bits of zeros. If two consecutive colons occur in a file that contains zero-byte records, they may be stored in the lower order portion of a word and create a zero-byte record.

Files created at a terminal under AUTO and TEXT commands or by using Text Editor contain zero-byte terminated records.

ZERO LENGTH PRU/RECORD

A PRU that contains no user data. If the level designator is zero, NOS calls it an EOR. CDC CYBER Record Manager calls it an EOR only for S type records. If the level designator is 17, NOS calls it an EOF and CDC CYBER Record Manager calls it end-of-partition. For a PRU device, COPYCF, COPYSBF, COPYX, and COPYBF copy to this boundary. Since a file can be subdivided into files by EOFs, the term multifile file arises in NOS.

6/7/8/9 MULTIPUNCH

Signifies an EOI on a card deck.

6/7/9 MULTIPUNCH

Signifies an EOF on a card deck.

7/8/9 MULTIPUNCH

Signifies an EOR on a card deck.

INDEX

AB 1-6-9
*A directive 1-C-4
A mode 1-8-3
Abnormal termination codes 1-B-1
Abort job 1-5-6; 1-10-3
Absolute binary cards 1-F-3, 4, 5
ACCESS 1-4-3
Access date 1-8-14
Access limits 1-6-2
Access word 1-6-8
Accessibility, tape 1-G-3, 7
Accessing files 1-2-9
Accessing direct access files 1-8-6
Access mode 1-8-2, 6, 7
Accessing tape files 1-10-13
Accessing unlabeled tapes 1-10-11
Account block SRU limit 1-6-18, 19
ACCOUNT statement 1-6-2
Accounting information 1-3-7
*ADD directive 1-C-4, 8
Address out of range 1-3-13; 1-6-12
Address registers 1-13-2
*AFTER directive 1-C-4, 6
Aging jobs 1-3-13
ALGOL statement 1-11-3
Alternate checkpoint dumps 1-10-17
Alternate system 1-1-2
Alternate user information 1-8-6, 7
Alternate user number 1-2-7; 1-8-2
ANSI labels 1-10-1, 9, 16; 1-G-1
ANSI labels
  End-of-file, end-of-volume coincidence 1-G-16, 17, 18
  Multifile, multivolume 1-G-15
  Multifile, single volume 1-G-14
  Single file, multivolume 1-G-13
  Single file, single volume 1-G-13
Answerback identifier 1-6-9
APPEND mode 1-8-3
APPEND statement 1-8-5
Appending information to a file 1-8-5
ARE 1-4-2
ARG= entry point 1-5-3
Arithmetic error 1-4-2
Arithmetic operators 1-4-1, 2
ASCII/display code conversion 1-10-2; 1-A-6, 7
ASCII mode 1-4-12
ASCII statement 1-4-12
ASCII terminals 1-F-6
Assembler languages 1-1-4
ASSIGN statement 1-7-2; 1-10-11
Assigning a file 1-10-11, 13
Assigning a pack 1-6-15
Assigning a tape unit 1-6-15; 1-10-17
Assigning equipment 1-10-11, 14, 17
Assigning nonallocatable devices 1-6-10
Assigning resources 1-6-15
ATTACH statement 1-8-6
Auto eject mode 1-F-5
Automatic permission 1-8-2
Auxiliary device requests 1-8-13
Auxiliary devices 1-2-8; 1-6-9; 1-8-4
Auxiliary devices, creating files on 1-6-10
AW 1-6-10

B format 1-10-27
*B directive 1-C-3, 4
Backspacing a file 1-7-3
Backup system 1-2-8
BASIC statement 1-11-6
BASIC subsystem 1-4-3
Batch jobs, submitting 1-6-21
Batch origin type 1-3-7
BATCH subsystem 1-4-3
BCD code 1-F-2
BCO 1-4-2
BCOT 1-3-7
*BEFORE directive 1-C-3, 4, 7
Beginning of information 1-2-2
Binary cards 1-F-3, 4, 5
Binary data 1-9-2
Binary punch output 1-2-4
Binary record management 1-C-1
Binary records 1-9-4, 5; 1-F-1
BKSP statement 1-7-3
Blank labeling a tape 1-10-12
BLANK statement 1-10-12
Block count 1-G-7, 9, 11
Block, defined 1-10-1
Blocked data format 1-2-2; 1-10-27
BOI 1-2-1, 2
Boolean operators 1-4-2
*BUILD directive 1-C-4, 9

CALL statement 1-4-5
Card deck 1-2-2
Card file structure 1-2-2
Card format 1-2-2; 1-F-2
Cards, binary 1-F-3, 4, 5

Cards, coded 1-F-2
Cards, punched 1-F-4, 5
Cards, read 1-F-2
Carriage control characters 1-A-8; 1-F-5
Carriage return delay 1-6-9
CATALOG statement 1-7-4
Catalog, user 1-8-2, 8
Category, file 1-8-2, 13
CATLIST statement 1-8-8
CB parameter 1-7-2; 1-12-3
CC 1-6-9
CCCCCCC 1-10-18
CCCCCCO 1-12-1
CEJ/MEJ option 1-1-1
Central exhange jump/monitor exchange jump 1-1-1
Central library directory 1-5-6
Central memory, defined 1-1-1
Central memory dumps 1-9-1, 2; 1-12-1; 1-13-1
Central memory field length 1-5-5
Central memory resident, defined 1-1-2
Central memory time slice 1-3-13
Central processor abort 1-3-14
Central processor priority, maximum 1-6-9
Central processor time 1-3-13; 1-5-5; 1-6-20
Central processor time, maximum 1-6-9, 20
Central processor unit, defined 1-1-1
CHANGE statement 1-8-10
Character conversion 1-F-4, 6
Character count 1-10-2
Character set conversion 1-10-2
Character sets 1-A-1; 1-F-6
Charge number 1-6-2,10
CHARGE statement 1-3-7; 1-6-2
Checkpoint dumps 1-10-13; 1-12-1
Checkpoint files 1-10-2; 1-12-1
Checkpoint/restart 1-12-1
Checksum 1-F-3, 5
CIO error messages 1-B-1
CK parameter 1-7-2; 1-12-3
CKP statement 1-12-1
CLEAR statement 1-7-7
CM data error 1-3-14
CM, defined 1-1-1; 1-6-9
CM dumps 1-13-1
CMC input error 1-3-14; 1-6-12
CM, maximum 1-6-9
CM time slice 1-3-13
CMR, defined 1-1-2
CN 1-6-10
COBOL statement 1-11-8
COBOL5 statement 1-11-11
Code, Hollerith 1-F-2
    Display 1-F-2
    Internal BCD 1-F-2
Coded cards 1-F-2, 4
Coded format 1-F-4
Coded records 1-F-2
*COMMENT directive 1-C-4, 9
COMMENT statement 1-6-3
*comment statement 1-6-3
COMMON statement 1-7-7
Communication area 1-B-1
Compare error 1-F-4
COMPASS source deck 1-3-2
COMPILE file 1-14-2, 6
Compiler languages 1-1-4
Completion of a job 1-3-15
Condition codes 1-3-13, 14
Configurations, PPU 1-1-3
Configurations, tape file 1-G-12
Constants 1-4-1
Continuation lines 1-10-10
Control language 1-4-1
Control language functions 1-4-8
Control language statements 1-4-4
Control point area 1-3-7
Control point dayfile 1-3-16
Control points, defined 1-1-2
Control statement format 1-5-1; 1-11-2; 1-C-2
Control statement input buffer 1-5-6
Control statement processing 1-3-8; 1-5-1
Control statement processing flow 1-5-6, 7
Control statements 1-5-1
    Checkpoint/restart 1-12-1
    File management 1-7-1
    Job control 1-6-1
    Load/dump central memory utility 1-9-1
    Loader control 1-5-1
    Permanent file 1-8-1
    Product set 1-11-1
    System utility 1-14-1
    Tape management 1-10-1
Conversion, character set 1-10-2
Conversion mode 1-5-6; 1-10-2; 1-F-2, 3
Conversion processes 1-F-2
CONVERT statement 1-7-7
*COPY directive 1-C-4, 10
COPY statement 1-7-8
COPYBF statement 1-7-9
COPYBR statement 1-7-10
COPYCF statement 1-7-10
COPYCR statement 1-7-11
COPYEI statement 1-7-12
COPYSBF statement 1-7-12
COPYX statement 1-7-13
Correction line images 1-9-3
Correspondence code terminals 1-F-6
CP directive 1-6-9
CPE 1-4-3
CPU, defined 1-1-1
CPU error exit 1-3-13

CPU hardware error exit mode 1-6-12
CPU priority 1-6-20
CPU program error exit mode 1-6-12
CPU programs 1-1-4
CPU time 1-6-6
CPU time limit 1-6-9
CPUMTR 1-1-2
Creating
    Direct access file 1-8-11
    Indirect access file 1-8-15, 16
    Labeled tape 1-10-13
    Library file 1-2-5; 1-C-1
    Tape files 1-10-1, 13
    Unlabeled tape 1-10-11
Creation date 1-8-14; 1-10-2; 1-G-6
Cross reference, system symbols 1-14-10
CS 1-6-9
CSET statement 1-4-12
CT option 1-8-2
CTIME statement 1-5-6; 1-6-3
Current time 1-6-18

*D directive 1-C-4
Data channels 1-1-3
Data conversion mode 1-F-2
Data format 1-10-3, 21
Data input 1-F-6
Data output 1-F-6
*DATA directive 1-C-4, 9
Dayfile messages 1-B-1
DAYFILE statement 1-6-3
DB 1-6-9
DC 1-7-31
DDP 1-1-3
Deadlock 1-6-13
DEBUG mode 1-6-7
Debugging aids 1-13-1
Deck structure 1-3-1
Deferred batch jobs 1-6-9
DEFINE statement 1-8-11
*DELETE directive 1-C-4, 7
Density, tape 1-10-3
Device, auxiliary 1-2-8
Device number 1-8-10, 14
Device, permanent file 1-2-8
Device residence 1-2-8
Device statistics 1-E-1
Device type 1-4-9; 1-7-2; 1-8-4
DF 1-6-9
Diagnostic messages 1-B-1
Direct access files 1-2-5, 7
    Accessing 1-8-6
    Block sizes 1-2-7; 1-E-1
    Changing parameters 1-8-10
    Defining 1-8-11
    Interlock 1-8-6
    Maximum size in PRUs 1-E-1
    Purging 1-8-14, 15
    Space 1-8-4
Directives
    Escape character 1-6-21
    MODIFY 1-14-2
    OPLEDIT 1-14-4
    UPDATE 1-14-6
Disable hardware exit mode 1-6-12
Disable program exit mode 1-6-12
Dismounting packs 1-6-15
Dismounting tapes 1-6-15
Display code 1-F-2
Display code dumps 1-9-2
DISPLAY statement 1-4-6
DISPOSE statement 1-7-14
Disposition of job output 1-6-21
Distributive data path 1-1-3
DMD statement 1-9-2
DMP statement 1-9-1; 1-13-1
DMP subroutine 1-13-2
DOCMENT statement 1-7-15
DS 1-6-10
Dump from a terminal 1-9-2
Dumping central memory 1-9-1, 2; 1-13-1
Dumps 1-13-1
Duplicate lines in dump 1-9-1; 1-13-1

E format 1-10-26
E mode 1-8-3
EBCDIC/display code conversion 1-10-2; 1-A-6, 7
EC 1-6-10; 1-7-32
EC directive 1-6-24
ECS 1-1-3
ECS data storage/retrieval 1-1-2
ECS files 1-2-2
ECS flag register operation parity error 1-3-13; 1-6-12
EDIT statement 1-14-1
Editing an OPL-formatted file 1-14-2
EF 1-4-3
EIO 1-4-3
EIOT 1-3-6, 7
E mode 1-8-3
EM 1-4-3
EM-M 1-13-2
EM-N 1-13-2
End of file 1-2-1
    B format 1-10-27
    E format 1-10-26
    F format 1-10-28
    I format 1-10-22
    S format 1-10-25
    SI format 1-10-24
    X format 1-10-24
End of file label 1-G-1, 8, 19

- End of information 1-2-1
  - B format 1-10-27
  - E format 1-10-26
  - F format 1-10-28
  - I format 1-10-22
  - S format 1-10-25
  - SI format 1-10-24
  - X format 1-10-25
- End of line byte 1-F-1
- End of record 1-2-1
  - B format 1-10-27
  - E format 1-10-26
  - F format 1-10-28
  - I format 1-10-22
  - S format 1-10-25
  - SI format 1-10-23
  - X format 1-10-24
- End of reel, defined 1-10-29
  - B format 1-10-27
  - E format 1-10-26
  - F format 1-10-28
  - I format 1-10-23
  - S format 1-10-25
  - SI format 1-10-24
  - Xformat 1-10-25
- End-of-tape, defined 1-10-29
- End of tape processing 1-10-29
- End-of-tape reflector 1-G-1
- End of volume label 1-G-1, 10, 19
- Enforce ring 1-10-6
- ENQUIRE statement 1-6-4
- EOF 1-2-1, 2
- EOF card 1-2-2; 1-3-1, 1-F-2
- EOF directive 1-6-22
- EOF1 label 1-G-1, 19
- EOI 1-2-1, 2
- EOI card 1-2-2; 1-3-1; 1-F-2
- EOR 1-2-1, 2
- EOR card 1-2-2; 1-3-1; 1-F-2
- EOR directive 1-6-22
- EOV1 label 1-G-1, 10
- EOV2 label 1-G-1
- EOV2-9 labels 1-G-1, 19
- Equipment/file assignment 1-10-11
- Error codes 1-B-1, 34
- Error conditions 1-3-13; 1-6-12, 13
- Error control 1-3-13; 1-6-7, 12; 1-13-2
- Error exit address 1-3-14
- Error exit mode 1-6-11, 12
- Error flag 1-3-14; 1-5-8; 1-6-12
- Error inhibit 1-10-7
- Error messages 1-B-1
- Error processing 1-5-8; 1-6-12
- Error processing bit 1-B-1
- Escape character 1-6-21
- Evaluation of control language expressions 1-4-4
- EVICT statement 1-7-16
- Exchange package 1-9-1; 1-13-1
- EXECUTE mode 1-8-3
- EXECUTE subsystem 1-4-3
- Exit mode 1-3-13
- Exit processing 1-5-8; 1-6-7, 12
- EXIT statement 1-3-14; 1-5-8; 1-6-7; 1-13-2
- Expiration date 1-G-6
- Export/Import origin type 1-3-6, 7
- Expressions 1-4-1
- Extended core storage 1-1-2
- External data format 1-2-2; 1-10-24
- External reference
  - ALGOL 4 1-11-3
  - BASIC 3 1-11-6
  - COBOL 1-11-8
  - COBOL5 1-11-11
  - FTN 1-11-17
  - SORTMRG 1-11-23

- F format 1-10-28
- Family 1-2-8
- Family device 1-8-4
- Family name 1-2-8; 1-6-5, 27
- FC 1-6-9; 1-7-32
- FET 1-B-1
- FET description 1-2-9
- FID 1-7-32
- Field length 1-3-8, 9; 1-5-5; 1-6-17; 1-13-2
- Field length assignment 1-3-8, 9; 1-5-5; 1-6-17
- Field length control 1-3-8, 9
- Field length, user defined 1-6-17
- File accessibility 1-2-9; 1-10-3
- File, block sizes 1-E-1
- File category 1-8-2
- File, checkpoint 1-12-1, 2
- File, communication area 1-2-9
- File, direct access 1-2-5; 1-8-6, 11
- *FILE directive 1-C-4, 5
- File environment table 1-B-1
- File function 1-4-8
- File header label 1-G-1, 4, 19
- File identifier 1-10-4; 1-G-5
- File management control statements 1-7-1
- File name, new 1-8-10
- File name table 1-2-3
- File password 1-8-2
- File permission mode 1-8-2, 14
- File, private 1-8-2
- File, public 1-8-2
- File, purging 1-8-14, 15
- File, replacing 1-8-15
- File, residency 1-8-11
- File, saving 1-8-16
- File section number 1-10-8; 1-G-2, 5
- File sequence number 1-10-8; 1-G-5

File function 1-4-8
File status 1-4-8
File status table 1-2-3
File, structure 1-2-1
File, types 1-2-3; 1-8-14
Files 1-2-1
Files, accessing 1-2-9
Files, local 1-2-4
Files, magnetic tape 1-10-1
Files, maximum number attached 1-6-9
Files, reading 1-2-9
Files, writing 1-2-11
First file header label 1-G-4
First end-of-file label 1-G-8
First end-of-volume label 1-G-10
First word address of memory 1-9-1
FL 1-4-3
FLE 1-4-3
FM 1-7-32
FNT entry 1-2-2,3
Force unload 1-10-7
Foreign data format 1-2-2; 1-10-28
Formats for cards read 1-F-2
Formats for cards punched 1-F-4
Formats for printed data 1-F-5
FORTRAN source deck 1-3-4
Frame count 1-10-3
FS 1-6-9
FST entry 1-2-3
FTN statement 1-11-17
FTNTS subsystem 1-4-3
FULL duplex transmission mode 1-6-9
Functions 1-4-2

Generation number 1-10-4; 1-G-6
Generation version number 1-10-3; 1-G-6
GET statement 1-8-12
gid 1-C-3
GOTO statement 1-4-4
Group record identifier 1-C-3
GTR statement 1-7-17

HALF duplex transmission mode 1-6-9
Hardware components 1-1-1
Hardware error exit mode 1-6-12; 1-13-2
Hardware instructions 1-1-4
HDR1 label 1-G-1,4
HDR2-9 labels 1-G-19
Header label, defined 1-G-1,4
Hollerith punch code 1-F-2,3
Hollerith punch output 1-2-4

I format 1-2-2; 1-10-22
*I directive 1-C-4
IC 1-7-33
ID 1-2-6; 1-7-33
IF statement 1-4-7
*IGNORE directive 1-C-4,7
Illegal instruction 1-3-13
Increasing the number of scheduled units 1-6-15
Increment registers 1-13-2
Indefinite operand 1-3-13; 1-6-12
Indirect access files 1-2-7; 1-8-15,16
- Accessing 1-8-6
- Appending information 1-8-5
- Block size 1-2-7; 1-E-1
- Changing parameters 1-8-10
- Creating 1-8-15,16
- Maximum number 1-6-9
- Maximum size in PRUs 1-6-9; 1-E-1
- Purging 1-8-14
- Replacing 1-8-15
- Saving 1-8-16

Infinite operand 1-3-12
INFT type files 1-2-3
Inhibit unload 1-10-6
Initiating a job 1-3-6
Input, data 1-F-6
INPUT file 1-3-12
Input file control 1-3-12
Input files 1-2-3
Input queue 1-3-8,13
Input queue priority 1-3-8
INPUT, writing on 1-3-12
*INSERT directive 1-C-4,6
Interchangeable families 1-2-8
Interlock 1-8-6
Internal BCD code 1-F-2
Internal data format 1-2-2; 1-10-22
IS 1-6-9

Job abort 1-5-6; 1-10-6
Job completion 1-3-15
Job control 1-3-8; 1-6-1
Job control statement 1-3-1; 1-5-4,5
Job control control statements 1-6-1
Job deck 1-3-1
JOB directive 1-6-21
Job field length 1-3-8,9; 1-5-5; 1-6-17
Job files 1-2-3
Job flow 1-3-1
Job initiation 1-3-6
Job name 1-3-6,7; 1-5-5, 1-6-7
Job name format 1-3-6,7
Job origin type 1-3-6
Job output information 1-D-1
Job priority 1-2-3; 1-3-8
Job scheduling 1-2-3; 1-3-8
Job statement format 1-5-4
Job step 1-3-8,9,12,31; 1-5-5; 1-6-20
Job structure 1-3-1
Job subsystem 1-4-3

Job termination 1-3-15
Jobs, submitting 1-6-21

Keywords 1-5-3
KRONREF statement 1-14-10

L format tapes 1-10-25
Label, defined 1-10-1
Label identifier 1-G-3,5,9,11,20
Label number 1-G-3,5,9,11,20
    Standard level 1-10-5; 1-G-3
LABEL statement 1-10-13
Label types 1-G-1
Labeled tape 1-10-8,9,13
Labels 1-G-1
Labels, optional 1-G-19
Labels, required 1-G-1
Last word address of memory 1-9-1
LBC statement 1-9-2
LDI statement 1-6-7
Legal user 1-6-2
LENGTH statement 1-6-8
LFM error codes 1-B-33
LGO 1-5-1; 1-12-1
LIBEDIT 1-C-1
LIBEDIT directive 1-C-1,3
LIBEDIT/LIBGEN examples 1-C-11
LIBEDIT statement 1-7-18; 1-C-1
LIBGEN statement 1-7-19
Library files 1-2-5; 1-C-1
Library, program 1-14-6
Library routines 1-11-1
Library type files 1-2-5,14
Library user 1-11-1
Libraries 1-2-12
    System 1-2-12
    Program 1-2-12
    User 1-2-12; 1-11-1
    User number LIBRARY 1-2-14
LIFT type files 1-2-5
LIMITS statement 1-3-7; 1-6-8
Line image data format 1-10-26
Line numbers 1-6-22
Line spacing 1-F-5
LISTLB statement 1-10-16
LIST80 statement 1-7-20
Literal input 1-F-3
Literals 1-5-2
Load map 1-13-3
Load/dump central memory utility control statements 1-9-1
Loaders 1-5-1
Loading binary data 1-9-2,4
Loading octal card images 1-9-3
Local file control statements 1-5-1
Local file, retrieve permanent file 1-8-12
Local file, saving 1-8-16
Local files 1-2-4
Location of data in core dump 1-13-4
LOC statement 1-9-3
LOCK statement 1-7-21
LOFT type files 1-2-4
Logical end of file 1-2-1
Logical end of information 1-2-1
Logical end of record 1-2-1
Logical/physical file structure 1-2-1
Logical record 1-2-1; 1-3-1
Long Block Stranger data format 1-10-25
LO72 statement 1-7-21
LP 1-6-9

M mode 1-8-3
Magnetic tape 1-10-1
Magnetic tape file, checkpoint 1-12-1,2
Magnetic tape file structure 1-2-2
Magnetic tape files, creating 1-10-1
Magnetic tape formats 1-2-2; 1-10-21
    Blocked 1-2-2; 1-10-27
    External 1-2-2; 1-10-24
    Foreign 1-2-2; 1-10-28
    Internal 1-2-2; 1-10-22
    Line image 1-2-2; 1-10-26
    System internal 1-2-2; 1-10-23
    Long blocked stranger tape 1-2-2; 1-10-25
    Stranger tape 1-2-2; 1-10-25
Magnetic tape labels 1-G-1
Magnetic tape units, maximum 1-6-9
Magnetic tapes, access restrictions 1-6-16
Magnetic tapes, assigning 1-6-15
Magnetic tapes, maximum 1-6-19
Magnetic tapes, scheduling 1-6-16
Magnetic tapes, system management 1-6-15
Main overlay 1-13-5
Managing tapes and packs 1-6-15
Mass storage device file structure 1-2-2
Mass storage device statistics 1-E-1
Mass storage files, size 1-8-4
Mass storage, maximum 1-6-10
Maximum field length 1-3-9; 1-5-5
Maximum number of control points 1-1-2
Maximum time 1-3-13
Memory boundaries 1-13-1
MEMORY macro 1-3-9
Memory, releasing 1-3-8; 1-6-16
Memory, requirements
    ALGOL 1-11-3
    BASIC 3 1-11-6
    COBOL 1-11-8
    FTN 1-11-17
    SORTMRG 1-11-23
MERGE file 1-14-6
Messages 1-B-1

MFL statement 1-6-11
MFL= entry point 1-3-8
MNE 1-4-3
MODE statement 1-3-14; 1-6-11
Modification date 1-8-14
Modification decks 1-14-4
Modification identifiers 1-14-4, 5
MODIFY statement 1-14-2
Modify directive 1-14-2
MODIFY mode 1-8-3
Modify-formatted program library file 1-14-4
MODVAL statement 1-6-11
MS 1-6-9
MT 1-6-9
Multifile reels 1-G-12
Multifile sets 1-G-1
Multireel files 1-G-12

N mode 1-8-3
NA option 1-8-5
*NAME directive 1-C-4, 5
ND option 1-8-5
Nested calls to procedure files 1-4-11
New file name 1-8-10
New password 1-6-14
New permanent file name 1-8-9
NEW statement 1-7-24
NEWPL file 1-14-6
NF 1-6-9
No abort option 1-8-5
No drop option 1-8-5
NOEXIT statement 1-3-14; 1-5-8; 1-6-13 1-13-2
Noise size 1-10-1, 5
Nonallocatable device 1-6-10
Nonallocatable devices, assigning 1-6-10
Nonstandard labels 1-G-1
NOPACK directive 1-6-22
*NOREP directive 1-C-4, 9
NORERUN statement 1-6-13
NORMAL mode 1-4-12
NOS labeled tape 1-10-4
NOS standard character set 1-A-4
NOS unlabeled tape 1-10-4
NOS written tape 1-10-1; 1-G-7
NOSEQ directive 1-6-22
NOTRANS directive 1-6-23
NPL file 1-14-4
NULL mode 1-8-3
NULL subsystem 1-4-3
NUM function 1-4-9
Number of jobs 1-6-8, 21

Octal card image format 1-9-3
Octal card images, loading 1-9-3
ODE 1-4-3
OF 1-6-9
OFA 1-10-5
OFFSW statement 1-6-13
Old password 1-6-14
OLD statement 1-8-12
OLDPL file 1-14-7, 9
ONEXIT statement 1-5-8; 1-6-13; 1-13-2
ONSW statement 1-6-14
Operand out of range 1-3-13; 1-6-12
Operand registers 1-13-2
Operating system format 1-5-1, 2, 3
Operators 1-4-1, 2
OPL file 1-14-2
OPL record type 1-7-5
OPLC record type 1-7-5
OPLD record type 1-7-5
OPLEDIT statement 1-14-4
Order-dependent format 1-5-3
Order-independent format 1-5-3
Origin type 1-3-6
Origin type, batch 1-3-7
Origin type, remote batch 1-3-7
Origin type, system 1-3-6
Origin type, time-sharing 1-3-7
OT 1-4-3
OUT statement 1-7-24
Output, data 1-F-6
Output file 1-2-4
OUTPUT file 1-2-4; 1-12-1
Output information 1-D-1
Overcommitment of resources 1-6-16
OVL record type 1-7-5
Owner identification 1-10-6, 1-G-3
Owner of a NOS written tape 1-10-1, 6
Owner of auxiliary device 1-2-8
O26 mode 1-5-6; 1-F-2
O29 mode 1-5-6; 1-G-3

PA 1-6-9
PACK directive 1-6-22
Pack name 1-2-8; 1-8-4, 13
PACK statement 1-7-25
PACKNAM statement 1-8-13
Packs, access restrictions 1-6-16
Packs, assigning 1-6-15
Packs, private 1-6-17
Packs, public 1-6-17
Packs, scheduling 1-6-15
Packs, shareable 1-6-17
Packs, system management 1-6-15

Paper tape reader 1-F-6
Parameter field 1-5-2
Parameters, control statement 1-5-3
Parameters, number of characters 1-5-3
PARITY statement 1-4-12
PASSWOR statement 1-6-14
Password 1-6-27; 1-8-2
Password, changing 1-6-10
PBC statement 1-9-4
PEE 1-4-3
Peripheral hardware 1-1-4
Peripheral processor library directory 1-5-6
Peripheral processor units 1-1-3
Permanent file control statements 1-8-1
Permanent file devices 1-2-8; 1-8-4, 1-E-1
Permanent file information 1-8-8
Permanent file name, changing 1-8-10
Permanent files 1-2-5
  Private 1-8-2
  Public 1-8-2
  Semiprivate 1-8-2
Permanent file size 1-E-1
Permanent file system 1-8-1
Permanent file request to auxiliary device 1-8-13
Permanent file, used as local file 1-8-12
Permanent files, direct access 1-2-7; 1-8-6
Permanent files, indirect access 1-2-7
Permanent files, purging 1-8-14, 15
Permission, file 1-8-2, 14
Permission information 1-8-7
PERMIT statement 1-8-14
PFM error codes 1-B-34
PFM error messages 1-B-1
PHFT type files 1-2-4
Physical file structure 1-2-1
Physical record unit, magnetic tape 1-2-2; 1-10-1
Physical record unit, mass storage 1-2-2
Physical record units, defined 1-2-1
PMFT type files 1-2-5
PN 1-6-10
PN option 1-8-4
PP record type 1-7-5
PPE 1-4-3
PPU 1-1-3
PPU record type 1-7-5
Prefix character 1-5-1
PRFT type files 1-2-4
PRI 1-7-33
PRIMARY statement 1-7-26
Primary terminal files 1-2-6
Print file priority 1-2-4
Print files 1-2-4
Printed data 1-F-5
Printer queue 1-2-4
Priority, CPU 1-6-5
Priority, job 1-2-3
Priority level 1-5-5
Priority, queue 1-3-8
Private file 1-8-2, 13
Private packs 1-2-8; 1-6-17
Procedure files 1-4-1, 10
Processing options 1-10-6
Product set control statements 1-5-1; 1-11-1
  ALGOL 1-11-3
  BASIC 1-11-6
  COBOL 1-11-8
  COBOL5 1-11-11
  FTN 1-11-17
  SORTMRG 1-11-23
Product set format 1-5-1, 2, 3
Program address 1-13-2
Program address register 1-9-1
Program call statements 1-11-2
Program error exit mode 1-6-12; 1-13-2
Program library 1-2-12
Program library utility control statements 1-14-1
Program name field 1-5-2
Project number 1-6-2, 10
PRU 1-2-1
  Magnetic tape 1-2-2; 1-10-1
  Mass storage 1-2-2
  Punched statements 1-2-2
PRU size 1-2-1; 1-E-1
PSE 1-4-3
Pseudo-sense switches 1-6-13, 26
PTFT type files 1-2-6
Public file 1-8-2
Public packs 1-2-8; 1-6-17
*PULLMOD directives 1-14-5
PUNCH 1-2-4
Punch code 1-F-4
Punch file structure 1-2-2
Punch files 1-2-4
PUNCHB 1-2-4; 1-9-4; 1-12-1; 1-F-5
Punched card format 1-F-4, 5
PURGALL statement 1-8-14
PURGE statement 1-8-15
Purging files 1-8-14, 15
PX 1-6-9
PW option 1-8-2
P8 1-2-4; 1-12-1; 1-F-5

Queue files 1-2-3; 1-7-14
Queue, input 1-3-13
Queue priority 1-3-8, 13
Queue, rollout 1-3-13
Queued file management 1-2-6
Queues, releasing files 1-7-14

R mode 1-8-3
R option 1-8-4
RA 1-1-2
RA mode 1-8-3
RA+1 1-1-2
Random access 1-2-9
RBR statement 1-9-4
READ directive 1-6-23
READ mode 1-8-3
READAP mode 1-8-3
Reading binary records 1-9-4
Reading CM dumps 1-13-3
Reading files 1-2-9
Reading statements 1-F-2
READMD mode 1-8-3
Record 1-2-1
Record logical 1-2-1
Record management 1-C-1
Record manager 1-11-1
Record prefix 1-9-4
Record type 1-7-5
Reference address 1-1-2; 1-13-2
Reference record identifier 1-C-3
Reflector, end-of-tape 1-G-1
Reformatting directive 1-6-22
REL type record 1-7-5
Relational operators 1-4-2
Releasing memory 1-3-8
Releasing output files 1-3-15, 16
Remote batch origin type 1-3-6, 7
Removable auxiliary devices, maximum, file residency 1-6-9; 1-8-4
Remove file 1-8-14, 15
RENAME statement 1-7-26
*RENAME directive 1-C-4, 10
REP 1-7-33
REPLACE statement 1-8-15
*REPLACE directive 1-C-4, 10
Replacing files 1-8-15
REQUEST statement 1-7-27; 1-10-17
Required tape labels 1-G-1
RERUN statement 1-6-14
Rerun status 1-6-14
Rescheduling a job 1-6-18
RESEQ statement 1-7-29
Reservation blocks 1-E-1
Residency, file 1-2-8; 1-8-9, 10
RESOURC statement 1-6-15
Resource types 1-6-15
Resource utilization 1-6-2
RESTART statement 1-12-2
Restarting a job 1-12-1, 2
Retention cycle 1-10-8
Retention date 1-10-8
RETURN statement 1-7-29
Returning a pack 1-6-17
Returning a tape file 1-6-17
REWIND directive 1-6-24
*REWIND directive 1-C-4, 5
REWIND statement 1-7-30
Rewrite in place 1-2-11
RFL= entry point 1-3-8
RFL statement 1-3-8; 1-6-17
rid 1-C-3
RM mode 1-8-3
RO 1-6-9
ROFT files 1-2-3
Rolling out a job 1-3-13
Rollout control 1-3-13
Rollout files 1-2-3; 1-3-13
Rollout queue 1-3-13
ROLLOUT statement 1-6-18
Rollout time period 1-3-13
ROUTE statement 1-3-6; 1-7-31
RP 1-6-9
RTIME statement 1-5-6; 1-6-18
Rubout characters 1-6-9
Running field length 1-3-8; 1-5-5
R1 1-4-3
R2 1-4-3
R3 1-4-3

S format 1-10-25
S option 1-8-4
Sample job 1-D-1
SAVE statement 1-8-16
Saving a file 1-8-16
SC 1-7-33
*SC directive 1-5-1
Scheduling jobs 1-2-3; 1-3-8
Scheduling packs 1-6-16
Scheduling resources 1-6-15
Scheduling tape units 1-6-15
Scheduling units 1-6-15
Scratch files 1-2-4
SDM= entry point 1-5-3
SECDED network 1-3-14
Section number 1-G-5
Security control 1-3-15
Security count 1-6-27
Semiprivate files 1-8-2
Sense switch 1-6-13, 26
Separators 1-5-2
SEQ directive 1-6-22
Sequence number 1-G-5
Sequential access 1-2-9
Set identification 1-G-5
Set identifier 1-10-8
SET statement 1-4-6
SETASL statement 1-6-18
SETCORE statement 1-6-19
SETID statement 1-7-34
SETJSL statement 1-6-19
SETPR statement 1-6-20
SETRFL macro 1-3-9
SETTL macro 1-3-12

SETTL statement 1-3-12; 1-6-20
Sharable packs 1-6-17
SI format 1-10-23
Single-error correction double-error detection network 1-3-14
SKIPEI statement 1-7-35
SKIPF statement 1-7-35
SKIPFB statement 1-7-35
SKIPR statement 1-7-36
SL 1-6-10
SORT statement 1-7-36
SORTMRG statement 1-11-23
SOURCE file 1-14-2, 7
Space for direct access file 1-8-4
Special control statements 1-5-6
Special files 1-2-4
SRE 1-4-3
SRU 1-3-7
SRU limit 1-3-12; 1-4-3; 1-6-10, 18
SS 1-4-3
ST 1-7-33
STAGE statement 1-7-38
Standard labels 1-G-1
Statement label field 1-5-1
Status information 1-B-1
STIME statement 1-5-6; 1-6-20
Stranger data format 1-10-25
Structure of files 1-2-1
SUBMIT statement 1-3-6, 7; 1-6-21
Submitting jobs 1-3-6; 1-6-21
Subroutine 1-13-6
Subsystem 1-4-3
SUI statement 1-6-25
SUMMARY statement 1-6-25
SWITCH statement 1-6-26
SYFT type files 1-2-6
symbolic names 1-4-2
SYO 1-4-3
SYOT 1-3-6
System code 1-G-7
System control statements 1-5-1
System default library 1-11-2
System description 1-1-1
System files 1-2-6
System internal data format 1-10-23
System job name 1-3-6
System library 1-2-12
SYSTEM macro 1-13-2
System monitor 1-1-2
System origin type 1-3-6
System origin privileges 1-6-9
System priorities 1-3-8
System resource units 1-3-7
System sequence number 1-3-6
System software 1-1-4
System utility control statements 1-14-1

Tape file configurations 1-G-12
Tape file structure 1-2-2
Tape formats 1-2-2
Tape labels 1-G-1
Tape management 1-10-1
Tape mark 1-10-1; 1-G-8, 10
Tape units
  Assigning 1-6-15
  Dismounting 1-6-17
  Scheduling 1-6-15
  System management 1-6-15
TAPEn 1-9-4, 5
Tapes, access restrictions 1-6-15
TC 1-6-9
TDUMP statement 1-7-39
TEFT file 1-2-3
Terminal character conversion 1-F-6
Terminal data input 1-F-6
Terminal parity 1-6-9
Terminal type 1-6-9
Termination 1-3-15
Terminators 1-5-2
TEXT record type 1-7-5
TID 1-2-6; 1-7-33
Time limit 1-3-10; 1-6-20
Time limit control 1-3-10; 1-6-20
Time limit error 1-5-8
Time of day 1-6-18
Time-sharing character set 1-A-1
Time-sharing commands 1-4-12
Time-sharing origin type 1-3-6
Timed/event rollout file 1-2-3
TKE 1-4-3
TL 1-6-9
TLE 1-4-3
TRANACT 1-4-3
TRANS directive 1-6-22
Transaction functions 1-6-10
Translate control statements 1-5-6
Transmission mode 1-6-9
Transparent mode 1-6-22, 23
TT 1-6-9
TTY character conversion 1-F-6
TXO 1-4-3
TXOT 1-3-6, 7
*TYPE directive 1-C-4, 5

UHLa labels 1-G-1, 20
ULIB record type 1-7-5
UN 1-7-33
UN option 1-8-2
Unlabeled tape 1-10-13,
Unload, force 1-10-7
Unload, inhibit 1-10-6
UNLOAD statement 1-7-40

UNLOCK statement 1-7-40
UPDATE statement 1-14-6
Update-formatted program library file 1-14-6
Update to modify conversion 1-14-9
UPMOD statement 1-14-9
USECPU statement 1-6-26
User catalog 1-8-2, 8
User header label 1-G-1, 20
User index 1-3-7; 1-6-25
User labels 1-G-20
User libraries 1-2-12; 1-11-1
User number 1-2-7; 1-3-7; 1-6-27
User number, alternate 1-8-2
User number library 1-2-14
User permission 1-8-3, 7
User programs 1-1-4
USER statement 1-6-27
User's control point dayfile 1-6-3, 4
User trailer label 1-G-1, 20
User validation 1-6-27
User volume label 1-G-1, 20
UTLa labels 1-G-1, 20
UVLa labels 1-G-1, 20

Validation 1-3-7
Validation information 1-6-8, 9, 10
VERIFY statement 1-7-41
VFYLIB statement 1-7-42
Volume accessibility 1-10-9
Volume, defined 1-10-1
Volume header label 1-G-1, 2
Volume serial number 1-10-1, 9
VOL1 1-G-1, 2, 3
VSN 1-G-3
VSN statement 1-10-18

W mode 1-8-3
WBR statement 1-9-5
Working files 1-2-4, 6
WRITE mode 1-8-3
WRITEF statement 1-7-42
WRITER statement 1-7-42
Writing files 1-2-11

X format 1-10-24

6/7/8/9 statement 1-2-2; 1-F-2
6/7/9 statement 1-2-2; 1-F-2
7/8/9 statement 1-2-2; 1-F-2
80-column binary punch output 1-2-4

COMMENT SHEET

MANUAL TITLE CDC NOS Version 1 Reference Manual, Volume 1

PUBLICATION NO. 60435400 REVISION C

FROM: NAME:

BUSINESS ADDRESS:

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

FOLD ON DOTTED LINES AND STAPLE

CUT ALONG LINE

STAPLE STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

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

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Publications and Graphics Division
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55112

CUT ALONG LINE

FOLD FOLD

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINNESOTA 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S

CONTROL DATA CORPORATION