60435400

CONTROL DATA CORPORATION

NOS VERSION 1
REFERENCE MANUAL

Volume 1 of 2

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

ALPHABETICAL LIST OF CONTROL STATEMENTS†

†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. Manual titles are listed in the preface. Refer to the NOS System Maintenance Reference Manual for a list of systems-oriented control statements.

60435400

CONTROL DATA CORPORATION

NOS VERSION 1
REFERENCE MANUAL

Volume 1 of 2

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

REVISION RECORD

REVISION	DESCRIPTION
A	Manual released. This manual reflects NOS 1.0 at PSR level 404.
(06-17-75)	
B	Revised to reflect NOS 1.1 at PSR level 419. New features include support of memory increments to
(03-08-76)	262K on 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	Revised to reflect NOS 1.2 at PSR level 439. New features include revised field length control, added
(12-03-76)	security for the CHANGE and PASSWOR control statements, queued file management, security count, SRU
	limit control, and additional parameters for the LIMITS statement. The parameters for the COBOL 5
	statement have been added to the product set descriptions. Four new control statements are
	described: MFL, ROUTE, SETASL, and SETJSL. New examples are included for creating multifiles on tape
	and using LIBEDIT. Technical and literary corrections have been made.
D	Revised to reflect NOS 1.2 at PSR level 452 and to make typographical and technical corrections. The
(07-15-77)	revision includes the TCOPY control statement, extensions to the COPY and VERIFY control statements,
	and support of the CYBER 171 computer system. In addition, the error messages in appendix B have been
	reformatted.
E	Revised to reflect NOS 1.2 at PSR level 460 and to make literary and technical corrections.
(11-21-77)	
F	Revised to reflect NOS 1.3 at PSR level 472. This revision adds descriptions of the following new
(05-26-78)	control statements: BEGIN, DMDECS, DMPECS, ENTER, NOTE, and PROTECT. The V carriage control
	character for programmable format is outlined. The new CYBER Control Language is presented with
	extensive use of examples. Section 11, Product Set Control Statements, is deleted. The product
	set control statement formats are given in the NOS Application Programmer's Instant. This edition
	obsoletes all previous editions.
G	Revised to reflect NOS 1.3 at PSR level 477 and to make literary and technical corrections.
(08-25-78)	
H	Revised to reflect NOS 1.3 at PSR level 485 and to correct literary and technical errors.
(12-22-78)	
J	Revised to reflect NOS 1.4 at PSR level 501. New features in this release include CYBER 170 Model 176
(08-10-79)	and 885 disk support; the FCOPY, HTIME, and TRMDEF control statements; and the 12-bit ASCII code set.
	This revision contains a new section 14, Library Maintenance, and a new appendix, Line Printer
	Carriage Control (I). This edition obsoletes all previous editions.

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.

REVISION RECORD (CONT'D)

REVISION	DESCRIPTION
K	Revised to reflect NOS 1.4. at PSR level 509. This revision contains the permanent file statement
(12-21-79)	parameters for the Mass Storage Facility (MSF). This edition obsoletes all previous editions.
L	Revised to reflect NOS 1.4 at PSR level 518. This revision contains the COPYL, COPYLM, and ITEMIZE
(05-23-80)	control statements; and a new appendix, Obsolete Tape Formats. This edition obsoletes all previous editions.
M	Revised to reflect NOS 1.4 at PSR level 530. This revision contains new UPDATE control statement
(12-05-80)	parameters to support the 12-bit ASCII code set, and revised OPLEDIT control statement list options compatible
	with the MODIFY control statement list options. Also included are the revised Procedures and Parameter
	Matching subsections in section 4, and the revised coded magnetic tape conversion tables in appendix A.

Publication No.
60435400

LIST OF EFFECTIVE PAGES

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

PAGE	REV
Front Cover	-
Inside Front Cover	M
Title Page	-
ii	M
ii-a/ii-b	M
iii	M
iv	M
v	M
vi	L
vi-a	M
vi-b	M
vi-c/vi-d	M
vii	M
viii	M
ix	M
x	M
xi	K
1-1-1	L
1-1-2	M
1-1-3	M
1-1-4	L
1-1-5	M
1-1-6	M
1-2-1	L
1-2-2	M
1-2-3	M
1-2-4	L
1-2-5	K
1-2-6	L
1-2-7	L
1-2-8	M
1-2-9	M
1-2-10	L
1-2-11	L
1-2-12	M
1-2-13	L
1-2-14	L
1-2-15	L
1-3-1	L
1-3-2	J
1-3-3	M
1-3-4	L
1-3-5	L
1-3-6	M
1-3-7	L
1-3-8	M
1-3-9	M
1-3-10	M
1-3-11	M
1-4-1	M
1-4-2	M
1-4-3	M
1-4-4	M
1-4-5	M
1-4-6	M
1-4-7	M
1-4-8	L
1-4-9	M
1-4-10	M
1-4-10.1/ 1-4-10.2	M
1-4-11	M
1-4-12	M
1-4-13	L
1-4-14	M
1-4-15	M
1-4-16	L
1-4-17	L
1-4-18	L
1-4-19	L
1-4-20	L
1-4-21	M
1-4-22	L
1-4-23	L
1-4-24	L
1-4-25	L
1-4-26	L
1-4-27	L
1-4-28	L
1-4-29	M
1-4-30	M
1-4-31	M
1-4-32	M
1-4-33	M
1-4-34	M
1-4-35	M
1-4-36	M
1-4-37	M
1-4-38	M
1-4-39	M
1-4-40	M
1-4-41	M
1-4-42	M
1-4-43	M
1-4-44	M
1-4-45	M
1-4-46	M
1-4-47	M
1-5-1	J
1-5-2	M
1-5-3	M
1-5-4	M
1-5-5	M
1-5-6	M
1-5-7	L
1-5-8	M
1-6-1	M
1-6-2	L
1-6-3	L
1-6-4	L
1-6-5	L
1-6-6	K
1-6-7	L
1-6-8	M
1-6-9	L
1-6-10	L
1-6-11	M
1-6-12	M
1-6-13	M
1-6-14	M
1-6-15	M
1-6-16	M
1-6-16.1/ 1-6-16.2	M
1-6-17	L
1-6-18	M
1-6-19	L
1-6-20	L
1-6-21	K
1-6-22	K
1-6-23	L
1-6-24	L
1-6-25	L
1-6-26	L
1-6-27	K
1-6-28	M
1-6-29	L
1-6-30	L
1-6-31	L
1-6-32	L
1-6-33	M
1-6-34	L
1-6-35	L
1-6-36	L
1-7-1	M
1-7-2	M
1-7-3	L
1-7-4	L
1-7-5	M
1-7-6	L
1-7-7	L
1-7-8	L
1-7-9	L
1-7-10	K
1-7-11	L
1-7-12	L
1-7-13	L
1-7-14	L
1-7-15	M
1-7-16	K
1-7-17	K
1-7-18	L
1-7-19	L
1-7-20	L
1-7-21	L
1-7-22	L
1-7-23	L
1-7-24	L
1-7-25	L
1-7-26	L
1-7-27	L
1-7-28	M
1-7-29	M
1-7-30	M
1-7-31	M
1-7-32	L
1-7-33	L
1-7-34	M
1-7-35	M
1-7-36	M
1-7-37	M
1-7-38	L
1-7-39	L
1-7-40	L
1-7-41	L
1-7-42	L
1-7-43	L
1-7-44	L
1-7-45	L
1-7-46	K
1-7-47	L
1-7-48	L
1-7-49	L
1-7-50	L
1-7-51	L
1-8-1	M
1-8-2	M
1-8-3	M
1-8-4	M
1-8-5	L
1-8-6	L
1-8-7	L
1-8-8	L
1-8-9	M
1-8-10	L
1-8-11	M
1-8-12	M
1-8-13	M
1-8-14	M
1-8-15	L
1-8-16	L
1-8-17	M
1-8-18	L
1-8-19	L
1-8-20	L
1-8-21	L
1-8-22	L
1-9-1	K
1-9-2	M
1-9-3	L
1-9-4	K
1-9-5	K
1-9-6	M
1-9-7	L
1-10-1	J
1-10-2	L
1-10-3	L
1-10-4	L
1-10-5	K
1-10-6	K
1-10-7	M
1-10-8	L
1-10-9	L
1-10-10	L
1-10-11	L
1-10-12	M
1-10-13	M
1-10-14	M
1-10-15	L
1-10-16	L
1-10-17	L
1-10-18	M
1-10-19	M
1-10-20	M
1-10-21	M
1-11-1	L
1-11-2	L
1-11-3	L
1-12-1	M
1-12-2	J
1-12-3	M
1-12-4	M
1-12-5	L
1-12-6	L
1-12-7	M
1-12-8	L
1-13-1	J
1-13-2	M
1-13-3	L
1-13-4	L
1-13-5	M
1-13-6	M
1-13-7	M
1-13-8	M
1-13-9	M
1-13-10	M
1-13-11	M
1-13-12	M
1-13-13	M
1-13-14	M
1-13-15	M
1-13-16	M
1-13-17	M
1-14-1	M
1-14-2	M
1-14-3	M
1-14-4	M
1-14-5	M
1-14-6	M
1-14-7	L
1-14-8	M
1-14-9	L
1-14-10	M
1-14-11	M
1-14-12	L
1-14-13	L
1-14-14	L
1-14-15	L
1-14-16	L
1-14-17	L
1-14-18	L
1-14-19	L
1-14-20	L
1-14-21	L

PAGE	REV
1-14-22	L
1-14-23	L
1-14-24	L
1-14-25	L
1-14-26	L
1-14-27	L
1-14-28	M
1-14-29	L
1-14-30	L
1-14-31	L
1-14-32	L
1-14-33	L
1-14-34	L
1-A-1	M
1-A-2	M
1-A-3	M
1-A-4	J
1-A-5	K
1-A-6	J
1-A-7	M
1-A-8	J
1-A-9	K
1-A-10	K
1-A-11	M
1-A-12	M
1-A-13	M
1-A-14	M
1-B-1	M
1-B-2	L
1-B-3	M
1-B-4	L
1-B-5	L
1-B-6	L
1-B-7	L
1-B-8	L
1-B-9	L
1-B-10	L
1-B-11	L
1-B-12	L
1-B-13	L
1-B-14	L
1-B-15	L
1-B-16	M
1-B-16.1/ 1-B-16.2	M
1-B-17	M
1-B-18	L
1-B-19	L
1-B-20	L
1-B-21	M
1-B-22	L
1-B-22.1/ 1-B-22.2	M
1-B-23	M
1-B-24	L
1-B-25	L
1-B-26	L
1-B-27	L
1-B-28	L
1-B-28.1/ 1-B-28.2	M
1-B-29	M
1-B-30	L
1-B-31	L
1-B-32	L
1-B-33	L
1-B-34	L
1-B-35	L
1-B-36	L
1-B-37	L
1-B-38	M

PAGE	REV
1-B-39	M
1-B-40	M
1-B-41	M
1-B-42	L
1-B-43	L
1-B-44	L
1-B-45	L
1-B-46	L
1-B-47	L
1-B-48	M
1-B-49	L
1-B-50	M
1-B-51	M
1-B-52	L
1-B-53	L
1-B-54	L
1-B-55	L
1-B-56	L
1-B-57	L
1-B-58	L
1-B-59	L
1-B-60	L
1-B-61	L
1-B-62	L
1-B-63	M
1-B-64	M
1-B-64.1/ 1-B-64.2	M
1-B-65	L
1-B-66	L
1-B-67	L
1-B-68	L
1-C-1	M
1-C-2	M
1-C-3	M
1-C-4	L
1-C-5	L
1-C-6	L
1-D-1	J
1-D-2	M
1-D-3	K
1-D-4	M
1-E-1	M
1-E-2	M
1-E-3	M
1-E-4	M
1-F-1	M
1-F-2	M
1-F-3	M
1-F-4	J
1-F-5	J
1-F-6	M
1-F-7	J
1-G-1	J
1-G-2	A
1-G-3	L
1-G-4	A
1-G-5	L
1-G-6	L
1-G-7	L
1-G-8	A
1-G-9	A
1-G-10	A
1-G-11	A
1-G-12	L
1-G-13	J
1-H-1	M
1-H-2	M
1-H-3	J
1-H-4	M
1-H-5	M

PAGE	REV
1-H-6	M
1-H-7	M
1-H-8	M
1-H-9	M
1-H-10	M
1-H-11	M
1-H-12	M
1-H-13	M
1-I-1	J
1-I-2	J
1-I-3	J
1-I-4	J
1-I-5	K
1-I-6	K
1-I-7	K
1-J-1	L
1-J-2	L
1-J-3	L
1-J-4	L
Index-1	M
Index-2	M
Index-3	M
Index-4	M
Index-5	M
Index-6	M
Index-7	M
Index-8	M
Index-9	M
Index-10	M
Index-11	M
Index-12	M
Index-13	M
Index-14	M
Index-15	M
Comment Sheet	M
Back Cover	-

PAGE	REV

PAGE	REV

PREFACE

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

AUDIENCE

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

Users should consult the glossary in appendix C for definitions of terms used in this manual.

ORGANIZATION

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

Volume 1 contains information for all NOS users. Included is a general description of the system and its handling of files and jobs, and detailed descriptions of control statement formats and processing. Appendixes include NOS character sets, messages, and a glossary.

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

CONVENTIONS

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

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

Extended memory for the CYBER 170 Model 176 is large central memory extended (LCME). Extended memory for all other NOS computer systems is extended core storage (ECS) or extended semiconductor memory (ESM).

In this manual, the acronym ECS refers to all forms of extended memory, unless otherwise noted.

Programming information for the various forms of extended memory can be found in the COMPASS Reference Manual and in the appropriate computer system hardware reference manual.

Program examples are written in the FORTRAN (Formula Translation) language.

RELATED PUBLICATIONS

The following is a list of NOS operating system manuals and NOS product set reference manuals. The NOS Manual Abstracts is a pocket-sized manual containing brief descriptions of the contents and intended audience of all NOS and NOS product manuals. The abstracts can be useful in determining which manuals are of greatest interest to a particular user.

Control Data also publishes a Software Release History Report of all software manuals and revision packets it has issued. This history lists the revision level of a particular manual that corresponds to the level of software installed at the site.

These manuals are available through Control Data sales offices or Control Data Literature Distribution Services (308 North Dale, St. Paul, Minnesota 55103).

Users requiring a list of the product control statements and their parameters should refer to the NOS Applications Programmer's Instant.

Control Data Publication	Publication Number
ALGOL Version 4 Reference Manual	60496600
ALGOL-60 Version 5 Reference Manual	60481600
APEX III Reference Manual	76070000
APL Version 2 Reference Manual	60454000
APT IV Version 2 Reference Manual	17326900
BASIC Version 3 Reference Manual	19983900
COBOL Version 4 Reference Manual	60496800
COBOL Version 4 to COBOL Version 5 Conversion Aid Version 1 Reference Manual	19265021
COBOL Version 5 Reference Manual	60497100
Common Memory Manager Version 1 Reference Manual	60499200
COMPASS Version 3 Reference Manual	60492600
Conversion Aids System Version 2 Reference Manual	19265358
CYBER Common Utilities Reference Manual	60495600
CYBER Database Control System Version 1 Reference Manual	60498700
CYBER Database Control System Version 2 Reference Manual	60481800
CYBER Interactive Debug Version 1 Reference Manual	60481400
CYBER Loader Version 1 Reference Manual	60429800
CYBER Record Manager Advanced Access Methods Version 2 Reference Manual	60499300
CYBER Record Manager Basic Access Methods Version 1.5 Reference Manual	60495700

Control Data Publication	Publication Number
CYBER 170 Computer Systems Hardware Reference Manual	60420000
CYBER 170 Computer Systems Models 720, 730, 750, 760, and 176 (Level B) Hardware Reference Manual	60456100
CYBER 70/Model 71 Computer System Hardware Reference Manual	60453300
CYBER 70/Model 72 Computer System Hardware Reference Manual	60347000
CYBER 70/Model 73 Computer System Hardware Reference Manual	60347200
CYBER 70/Model 74 Computer System Hardware Reference Manual	60347400
Data Base Utilities Version 1 Reference Manual	60498800
Data Catalogue 2 Reference Manual	60483200
DDL Version 2 Reference Manual, Volume 1	60498400
DDL Version 2 Reference Manual, Volume 2	60498500
DDL Version 2 Reference Manual, Volume 3	60498600
DDL Version 3 Reference Manual, Volume 1	60481900
DDL Version 3 Reference Manual, Volume 2	60482000
DDL Version 3 Reference Manual, Volume 3	60482100
Export/Import Reference Manual	60436200
FORM Version 1 Reference Manual	60496200
FORTRAN Common Library Mathematical Routines Reference Manual	60498200
FORTRAN Data Base Facility Version 1 Reference Manual	60482200
FORTRAN Extended Version 4 Reference Manual	60497800
FORTRAN Extended Version 4 to FORTRAN Version 5 Conversion Aids Program Version 1 Reference Manual	60483000
FORTRAN Version 5 Common Library Mathematical Routines Reference Manual	60483100
FORTRAN Version 5 Reference Manual	60481300
General Purpose Simulation System V (GPSS) General Information Manual	84003900
Interactive Graphics System Application Executive Reference Manual	17322200
Message Control System Version 1 Reference Manual	60480300

Control Data Publication	Publication Number
Modify Version 1 Instant Manual	60450200
Modify Version 1 Reference Manual	60450100
Network Products Communication Control Program (CCP) Version 3 Reference Manual	60471400
Network Products Interactive Facility Version 1 Reference Manual	60455250
Network Products Interactive Facility Version 1 User's Guide	60455260
Network Products Network Access Method Version 1 Network Definition Language Reference Manual	60480000
Network Products Network Access Method Version 1 Reference Manual	60499500
Network Products Network Terminal User's Instant	60455270
Network Products Remote Batch Facility Version 1 Reference Manual	60499600
Network Products Stimulator Version 1 Reference Manual	60480500
Network Products Transaction Facility Version 1 CYBER Record Manager Data Manager Reference Manual	60456710
Network Products Transaction Facility Version 1 Data Manager Reference Manual	60455350
Network Products Transaction Facility Version 1 Reference Manual	60455340
Network Products Transaction Facility Version 1 User's Guide	60455360
NOS Version 1 Application Installation Handbook	84000970
NOS Version 1 Applications Programmer's Instant	60436000
NOS Version 1 Batch User's Guide	60436300
NOS Version 1 Diagnostic Index	60455720
NOS Version 1 Installation Handbook	60435700
NOS Version 1 Manual Abstracts	84000420
NOS Version 1 Operator's Guide	60435600
NOS Version 1 Reference Manual, Volume 2	60445300
NOS Version 1 System Maintenance Reference Manual	60455380
NOS Version 1 Systems Programmer's Instant	60449200

Control Data Publication	Publication Number
NOS Version 1 Terminal User's Instant Manual	60435800
NOS Version 1 Time-Sharing User's Guide	60436400
NOS Version 1 Time-Sharing User's Reference Manual	60435500
On-Line Maintenance Software Reference Manual	60454200
PERT/Time Version 2 Reference Manual	60456030
PL/I Version 1 Reference Manual	60388100
Query Update Version 3 Reference Manual	60498300
SIMSCRIPT Version 3 Reference Manual	60358500
Software Publications Release History	60481000
Sort/Merge Versions 4 and 1 Reference Manual	60497500
SYMPL Version 1 Reference Manual	60496400
TAF/TS Version 1 CYBER Record Manager Data Manager Reference Manual	60456700
TAF/TS Version 1 Data Manager Reference Manual	60453100
TAF/TS Version 1 Reference Manual	60453000
TAF/TS Version 1 User's Guide	60436500
Text Editor Version 1 Reference Manual	60436100
TOTAL-CDC Reference Manual	76070300
Update Version 1 Reference Manual	60449900
XEDIT Version 3 Reference Manual	60455730
6400/6500/6600 Computer Systems Hardware Reference Manual	60100000
8-Bit Subroutines Reference Manual	60495500

DISCLAIMER

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

CONTENTS

VOLUME 1

APPENDIXES

INDEX

FIGURES

TABLES

VOLUME 2

APPENDIXES

SYSTEM DESCRIPTION 1

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

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

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

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

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

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

SYSTEM HARDWARE

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

- Central processor unit(s).
- Central memory.
- Extended memory (optional).
- Peripheral processors.
- Peripheral equipment.

CENTRAL PROCESSOR UNIT

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

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

CENTRAL MEMORY

The primary functions of central memory (CM) are:

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

Job Field Length

The job field length is the portion of central memory that is assigned to a user's job. Several jobs can reside in CM simultaneously. The field length separates each job while it resides in CM. The job is assigned a starting CM address (its reference address or RA) and allocated an initial field length (the CM words in which the job resides and executes). The field length is adjusted during job execution as described in section 3. Figure 1-1-1 shows a job field length within CM.

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

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

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

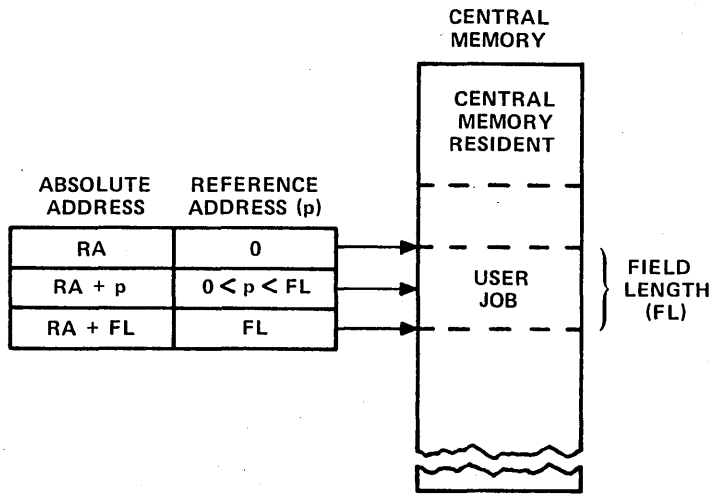


Figure 1-1-1. Central Memory Allocation

Central Memory Resident

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

EXTENDED MEMORY

Extended memory for the NOS computer systems is called ECS (refer to preface). ECS can be used for the following purposes.

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

Only validated users can use extended memory (refer to LIMITS Statement in section 6).

†CYBER 170 Model 176 extended memory cannot link mainframes.

PERIPHERAL PROCESSORS

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

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

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

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

PERIPHERAL EQUIPMENT

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

405 Card Reader

415 Card Punch

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

844-21 Disk Storage Subsystem

844-41 and 844-44 Disk Storage Subsystems

885 Disk Storage Subsystem

Mass Storage Facility (MSF)

667, 669, 677, and 679 Magnetic Tape Units

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

6671 or 6676 Multiplexers for communication with interactive terminals

255x Network Processing Units

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

SYSTEM SOFTWARE

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

USER PROGRAMS

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

- Compiler languages provide the user with a language suited to his particular needs. The program statements are translated by the appropriate compiler [FORTRAN, COBOL (Common Business-Oriented Language), ALGOL (Algorithmic Language), and so on], which 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, normally used by advanced programmers, are machine-dependent. Most of the NOS system is written in COMPASS, the assembler language of the CYBER 170, CYBER 70, and 6000 Series computers.

- Hardware instructions are interpreted directly by the computer and, thus, 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 except when program debugging requires that the user interpret memory dumps.

OPERATING SYSTEM

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

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

CYBER Loader

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

CYBER Record Manager

CYBER Record Manager (CRM) is the interface between user input/output (I/O) functions and NOS physical I/O functions. The NOS operating system control statements do not use CRM. Some of the products that use CRM are COBOL 4, COBOL 5, FORTRAN Extended 4, FORTRAN 5, Sort/Merge 4, ALGOL 4, ALGOL 5, PL/I (Programming Language I), and DMS-170.

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

FILES 2

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

FILE NAMES

Each file has a unique one- to seven-alphanumeric-character name.†

Examples:

```
A  123  TAPE  1A2B  COMPILE
```

NOTE

If all the following conditions are true, the seventh character of a file name does not make the name unique, and NOS assumes that the tape file is the file referenced. The conditions are:

- A tape file with a six- or seven-character name is assigned to the job.
- The job references a six- or seven-character name of a file that does not exist.
- The first six characters of the file names match.

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

```
SCR  SCR1  SCR2  SCR3  SCR4
```

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

```
RESERVED FILE NAME.
```

†Some products such as FORTRAN Extended 4, FORTRAN 5, and COBOL 5 do not support file names that begin with a digit. Also, some products support only six-character file names. Refer to the product reference manual listed in the preface for details.

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

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

INPUT OUTPUT PUNCH PUNCHB P8

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

FILE STRUCTURE

File structure within a computer system has several meanings. The NOS user can think of a file as having three representations; two logical representations (CYBER Record Manager file structure and NOS file structure) and a physical representation. Logical file structure is how the user orders the data. The user can define this logical file structure using higher level language statements within a source program. CYBER Record Manager (CRM) translates the higher level language statements into the file structure that it superimposes on the data. NOS file and record marks structure a file while it is being processed within the system. NOS converts the NOS file and record marks to their physical tape, disk, or card equivalents when the file is stored.

CYBER RECORD MANAGER FILE STRUCTURE

CYBER Record Manager handles I/O for several products (refer to section 1) including FORTRAN Extended 4, FORTRAN 5, and COBOL 5. CRM superimposes its file structure on the NOS file structure. Through CRM, the user can specify a file organization, a blocking type, and a record type for his data. The file organization determines how records are accessed, the blocking type determines how CRM records are grouped on their storage media, and the record type defines the smallest unit of data CRM can retrieve. This manual does not describe CRM. A description of CRM including the FILE control statement is included in the CRM manuals listed in the preface.

NOS FILE STRUCTURE

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

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

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

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

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

The last EOF in a file may or may not be present depending upon the program used to create the file.

PHYSICAL FILE STRUCTURE

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

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

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

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

†For more information, refer to appendix F.
††The 6/7/9 card is not recognized in a remote batch job. In an RBF job the end-of-file marker is a card with a 7/8/9 punch in column 1 and a level number of 17_8 in columns 2 and 3. RBF/HASP can also use a card with /*EOR in columns 1 through 5 and a level number of 17_8 in columns 6 and 7.
†††For more information, refer to section 10 and appendix G.

Card Files

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

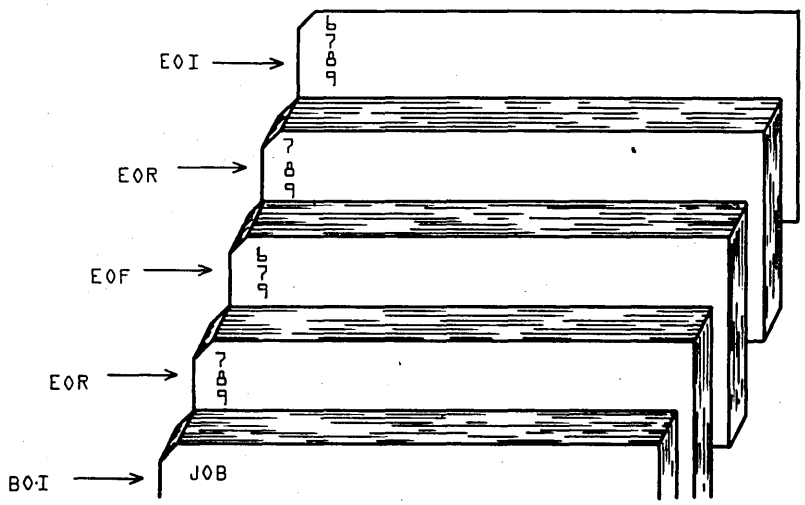


Figure 1-2-1. Sample Card File Structure

Mass Storage Files

Mass storage files are stored on disk or ECS.

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

† Keypunch mode selection is not supported for jobs entered through a 200 User Terminal or similar remote batch terminal except for HASP.

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

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

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

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

Magnetic Tape Files

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

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

Tape Labels

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

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

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

The ANSI label EOF indicates end-of-information for a file within a file set. The use of ANSI labels to delimit files within file sets is illustrated in figure 1-2-2.

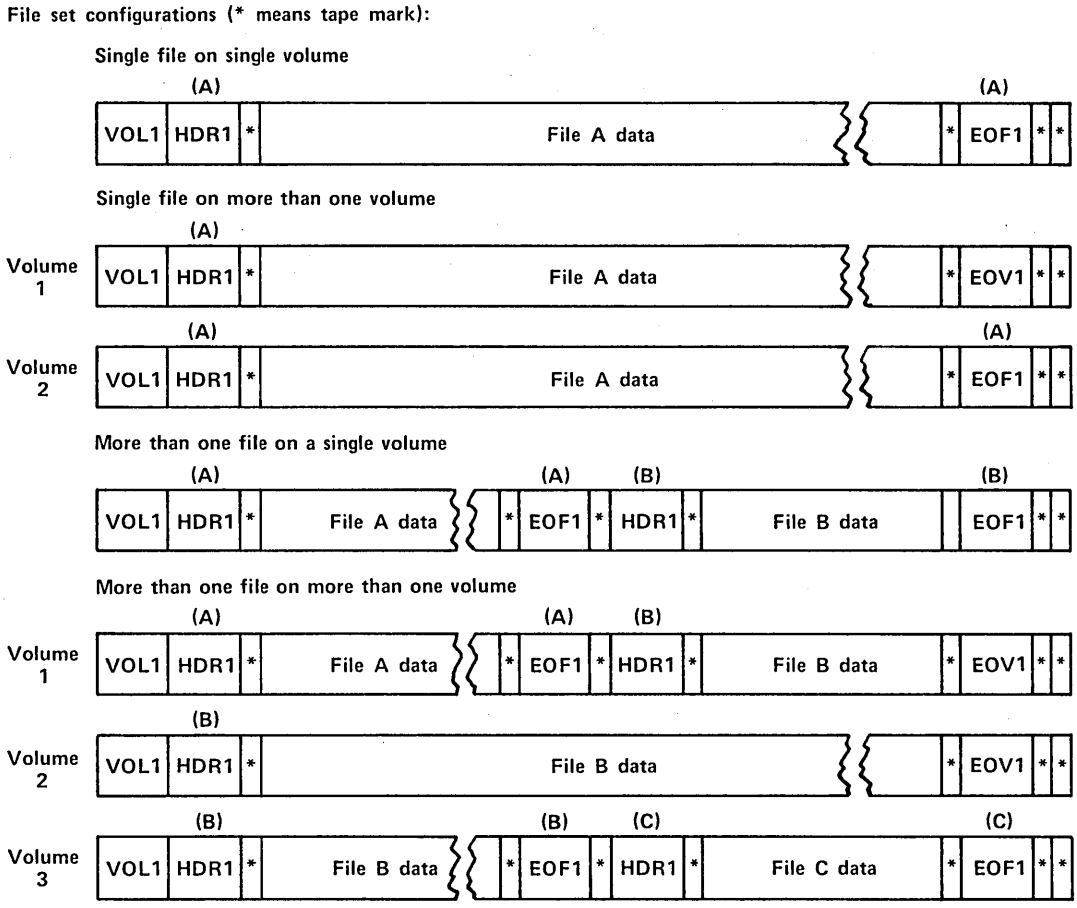


Figure 1-2-2. Use of ANSI Labels

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

Label	Location
VOL1	Beginning of volume.
HDR1	Beginning of information. If the file continues on to another volume, the HDR1 label is repeated. It must follow the VOL1 label and precede the continuation of the file information.
EOF1	End of information.
EOV1	End of volume (required only if the file continues on another volume).

Appendix G gives the tape label formats.

Tape Data Formats

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

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

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

Tape Format	Labels	Recording Mode	Noise Size††	
I	Labeled or unlabeled	Binary only	Seven-track:	< eight frames
			Nine-track:	< six frames
SI	Labeled or unlabeled	Binary only†††	Seven-track:	< eight frames
			Nine-track:	< six frames
S	Labeled or unlabeled	Binary or coded	User-selected; default is	< 18 frames

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

Tape Format	Labels	Recording Mode	Noise Size	
L	Labeled or unlabeled	Binary or coded	User-selected; default is	< 18 frames
F	Unlabeled (labels read as data)	Seven-track: binary or coded; Nine-track: binary only†	User-selected; default is	< 18 frames

NOS terminates blocks on I and SI format tapes with a 48-bit block (PRU) terminator. The terminator contains the total number of bytes in the block (including the terminator itself), the number of blocks since the last HDR1 label, and the level number of the block. This terminator enables read operations on I format tapes to check whether the number of bytes read and the block number expected match the byte count and block number in the terminator. If either does not match, the system attempts to recover the missing data. This feature prevents dropped or fragmented blocks and provides a higher degree of reliability than other data formats.

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

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

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

FILE TYPES

Every file assigned to a user's job has a file type. A file assigned to a job is known to the system by its entry in the file name table/file status table (FNT/FST). An FNT/FST entry contains the file name, the device on which the file resides, the file type, and its current position and status.

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

†No code conversion is performed even if coded data is read.

FILES ASSIGNED TO USER JOBS

NOS uses the following mnemonics to define file types.

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

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

Input Files

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

Print Files

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

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

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

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

---

†Not applicable to time-sharing jobs.

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

Print files must be formatted for line printing. The user should add appropriate printer control characters (refer to appendix I, Line Printer Carriage Control). Appendix D contains the printer output from the compilation and execution of a sample program.

Punch Files

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

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

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

Local Files

Local files are temporary files. The local file type includes all scratch copies of files except the primary file.

The user can create a local file by:

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

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

Primary Files

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

Direct Access Files

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

Library Files

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

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

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

   LOCK(lfn)

   COMMON(lfn)

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

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

Rollout Files

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

Timed/Event Rollout Files

A timed/event rollout file is similar to a rollout file in that it contains all the information concerning a job temporarily removed from central memory. However, a timed/event rollout file is rolled back into central memory only when a specified event has occurred (such as a file becoming not busy or a tape being mounted) or a specified time period has elapsed.

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

PERMANENT FILES

Permanent files are retained on mass storage until their creator purges them. Each permanent file has a permanent file catalog entry associated with a user number. Each permanent file catalog describes all permanent files created under that user number and their location on mass storage. Unless another user number is specified, the system assumes that all permanent file requests are made to the catalog of the user number named on the last USER statement (or named in the login of a time-sharing job).

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

- UABCDEF
- UABDDEE
- MABCDE1

The device residence of permanent files and their backup copies are described under Mass Storage File Residence in this section.

The two types of permanent files, indirect access permanent files and direct access permanent files, are described in the following paragraphs.

Indirect Access Permanent Files

The user accesses an indirect access permanent file by naming it in an OLD or GET control statement. The system copies the permanent file from mass storage to a temporary file (local or primary file type). To alter an indirect access permanent file the user makes the changes to the temporary copy and then enters the REPLACE control statement which writes the temporary copy over the indirect access permanent file. The user creates an indirect access permanent file by naming a temporary file in a SAVE or REPLACE control statement.

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

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

Direct Access Permanent Files

The user accesses a direct access permanent file directly, not through a temporary copy. Data is written directly on the permanent file.

The user creates a direct access permanent file with a DEFINE control statement, which determines its name and residence and the default access mode available to other users. He accesses the file with an ATTACH control statement. A number of users can attach the file concurrently, but only one user at a time can change the file. To change the file, the user must attach it in modify, append, or write access mode. If a user attaches the file in write mode, no other user can attach that file concurrently.

Even if a file is currently attached to a job, the user can purge the file from the permanent file catalog with a PURGE statement. However, the purged direct access file remains attached to the job until it is released by a RETURN, CLEAR, UNLOAD, OLD, or NEW statement or until the job ends.

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

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

MASS STORAGE FILE RESIDENCE

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

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

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

FAMILY DEVICES

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

Normally, a system has only one family (the default family) of permanent file devices. However, because families are interchangeable between NOS systems, several families may be active on one system, or a system may be part of a multimainframe system. For example, consider an installation with two systems, A and B. System B provides backup service to system A. 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 one- to seven-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 the user's family is the system's default family, the user may, but need not, supply the 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 files on family devices, he has the option of either using the system default device type or specifying another type of permanent file device.

AUXILIARY DEVICES

An auxiliary device is a supplement to the mass storage provided by family devices. It is identified by a one- to seven-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 or replace those files as permitted by the owner.

MASS STORAGE FACILITY (MSF)

Magnetic disk is the usual residence of permanent mass storage files. However, if the installation has an MSF, some direct access files can be stored there. An MSF is suited for the storage of large, direct access files that are accessed infrequently. Attaching a file residing on the MSF takes at least 10 seconds, because the file must be retrieved and copied (staged) to disk. Users can specify the preferred residence of their direct access files with the PR parameter on the DEFINE or CHANGE statements. They can determine the actual residence of their files with the CATLIST, LO=F statement.

Usually, when attaching an MSF file, the system rolls out the job until the file has been staged and assigned to the job. (The time-sharing user can determine the status of his MSF file staging request with the ENQUIRE statement.) However, if the user specifies the RT parameter on the ATTACH statement, the job continues processing while the MSF file is being staged to disk. The user must then issue a second ATTACH statement to assign the file to his job after staging. The user can then check that the file has been attached by using either a FILE function (in a batch job or procedure file) or a LENGTH or ENQUIRE(F) statement (in a time-sharing job).

If a permanent file is lost or destroyed, site personnel can recover the file by loading its backup copy. Generally, sites perform regular dumps of permanent files to magnetic tape to serve as the permanent file backup. By specifying the BR parameter on the DEFINE or CHANGE statement, the user can choose to have a tape backup copy of his direct access file kept even if the file resides on the MSF. He also can choose to have the MSF file copy serve as backup, or he can require no backup copy of his direct access file.

LIBRARIES

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

USER NUMBER LIBRARY

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

PROGRAM LIBRARIES

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

USER LIBRARIES

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

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

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

JOB FLOW AND EXECUTION 3

A job is a file of statement images.† Its first record contains control statements that specify job processing requirements. Every job begins with a job statement and a USER statement. The end of the control statement record is marked by an EOR (or an EOI if there is no data in the job).

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

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

JOB INITIATION

The user initiates jobs by:

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

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

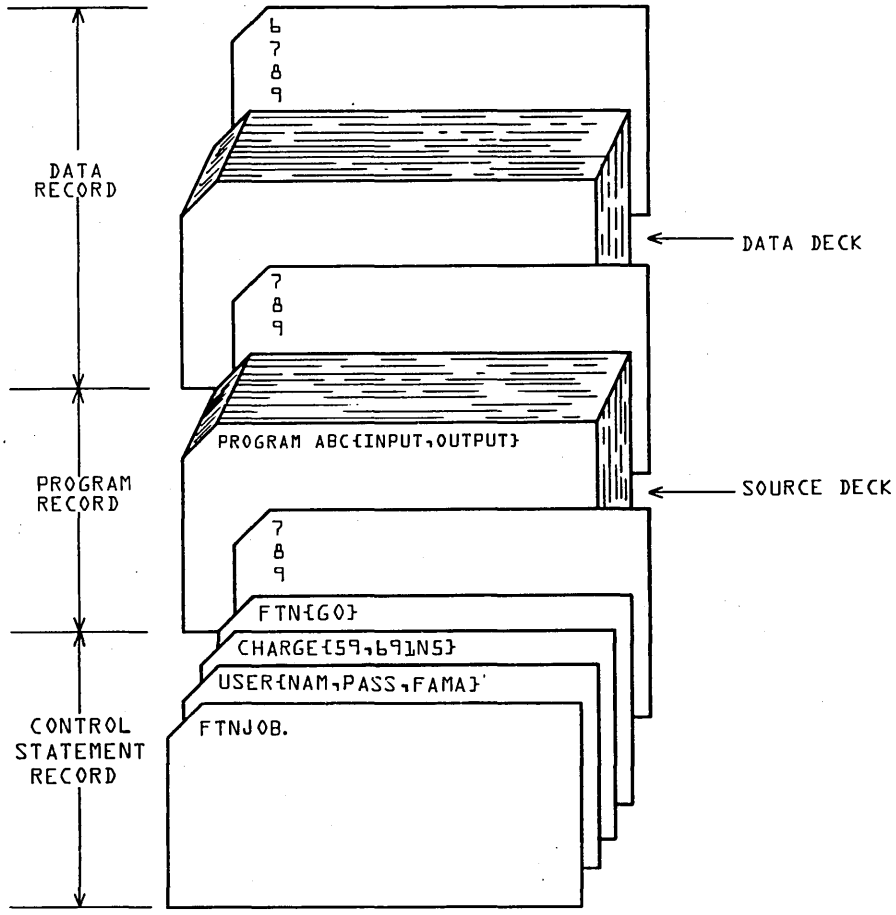


Figure 1-3-1. FORTRAN Compile and Execute Deck

JOB ORIGIN TYPES

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

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

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

JOB NAMES

After entering the system, the job is assigned a unique seven-character job name to prevent job name duplication within the system. This name is not the job name specified on the job statement. The job name appears on the banner page of all batch output printed by the job.

SYSTEM JOB NAME FORMAT

The first four characters of a system job name are obtained from the job name entered or are display code zero-filled if fewer than four characters are entered. The last three characters are a unique system job sequence number in the range from AAA to ZZZ. For example, if the job entered is DIS, a possible job name is DIS0AAB.

LOCAL BATCH AND RBF JOB NAME FORMAT

The first four characters of a local batch or RBF 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 last three characters are the job sequence number.

TIME-SHARING, IAF, AND EXPORT/IMPORT 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 last three characters represent the connection number of the terminal on which the user is logged in.

DEFERRED BATCH JOB NAME FORMAT

All jobs entered via a ROUTE, SUBMIT, or LDI control statement derive the first four characters of their job names from the job's current user index. For all deferred batch jobs originating from system, local batch, Export/Import, and RBF jobs, the last three characters are the system job sequence number.

VALIDATION

The USER statement follows the job statement and is used to validate the user as a legal user (refer to USER Statement in section 6). If the user is validated, a set of control values is associated with the job; these values are used by the system to control all system requests from the job. If the user is not permitted to perform specific functions (such as access nonallocatable devices), the user's job is aborted and a message such as

ILLEGAL USER ACCESS.

is issued when the illegal function is attempted.

To determine the extent of his validation, the user can issue the LIMITS control statement and receive a listing of his current validation control values. Refer to LIMITS 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. The system uses this index to determine the location of the user's permanent file catalog. (Refer to the NOS System Maintenance Reference Manual for an explanation of the user index.)

ACCOUNTING

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

- Central processor time.
- Mass storage activity.
- Adder activity (fixed charges for some highly variable system requests).
- Magnetic tape activity.
- Permanent file activity.
- SRU value.
- Application account charges.†

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

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 there are changes (for example, when the field length of a job is released, a job enters a queue, or a job completes).

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.

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

JOB CONTROL

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

FIELD LENGTH CONTROL

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

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

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

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

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

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

- Predefined initial field length for a system routine (RFL= or MFL= special entry point as described in appendix F, volume 2).
- Highest high address (HHA) from EACP loader table (54 table) (refer to the CYBER Loader Reference Manual).
- RFL value, if nonzero.
- The smaller of the MFL or the installation-defined default value (release value 50000B).

NOTE

The system automatically assigns a field length for CM only. To set the initial field length for a job step in ECS, the user must use the RFL statement or the MEMORY macro (refer to volume 2).

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

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

Control Statement	MAXFL	MFL	RFL	Field Length	Explanation
JOB(CM60000) USER(USERABC,1234,FAM1) CHARGE(4922,66X)	60 000 60 000 60 000	60 000 60 000 60 000	0 0 0	700 700 2 200	The CM parameter sets the MAXFL and MFL values. The system sets the field length as required for processing the control statements.
GET(ABSPROG,RELPROG)	60 000	60 000	0	1 700	GET statement retrieves copies of an absolute program and a relocatable program.
RFL(40000)	60 000	60 000	0	1 500	The user issues an RFL statement to set the field length for execution of the absolute program that follows.
ABSPROG.	60 000	60 000	40 000	40 000	The absolute program on file ABSPROG is executed within a 40 000-word field length.
MFL(50000)	60 000	60 000	40 000	1 500	The user issues an MFL statement to set the maximum field length for the following relocatable load.
RELPROG.	60 000	50 000	0	≤ 50 000	If more than a 50 000-word field length is required, the job aborts.

INPUT FILE CONTROL

All user jobs, when initiated, have a file named INPUT (INFT type file). This file contains the control statements and other input records required for job execution. INPUT is a locked file. As a result, the user may read from it and reposition it, but the system does not allow him to write on it. If for some special reason the user needs to write on INPUT, he should first issue a RETURN(INPUT) control statement (refer to section 7). This statement changes the name of the file from INPUT to INPUT* and leaves it assigned to the user's job. The user may then write on file INPUT. The change of name on RETURN applies only if the input file is of type INFT (refer to File Types in section 2).

TIME LIMIT CONTROL

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

While a job is using the central processor, the CPU time 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 Network Product Interactive Facility Reference Manual or the NOS Time-Sharing User's Reference Manual for more details.

SRU LIMIT CONTROL

The system sets a limit on the number of system resource units (SRU) that a job step or an account block can accumulate. An SRU includes central processor time, central memory usage, permanent file activity, and mass storage and tape I/O. An account block is that portion of a job from one CHARGE statement to the end of the job or to 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, SRUs are accumulated and checked against the SRU step and account block limits. If the job is not a time-sharing job (TXOT), the job is aborted when either limit is reached. Time-sharing jobs are rolled out. After a time-sharing job is rolled out, the user can increment the limit and resume execution from the point where the limit was reached. Refer to the Network Products Interactive Facility Reference Manual or the NOS Time-Sharing User's Reference Manual for more details.

CONTROL STATEMENT LIMIT CONTROL

If a job attempts to execute more control statements than the number for which the user is validated, the following message is issued.

INITIAL CONTROL STATEMENT LIMIT.

NOS then searches for an EXIT statement. If it does not find one, it terminates the job immediately. If it does find an EXIT statement, it allows processing of seven additional control statements for job error processing. After processing the seven additional statements, NOS terminates the job after issuing the following message.

CONTROL STATEMENT LIMIT.

A user can determine his control statement limit validation by entering a LIMITS statement (section 6).

ROLLOUT CONTROL

Each executing program is allowed to reside in CM for a certain amount of time before relinquishing its space to another program. When this CM time slice is exceeded, the program may be rolled out. This means that the contents of the job field length (both CM and ECS), 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; 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; queue priorities can also vary for each origin type. The queue priority is aged (incremented) while the job is in the rollout queue. Normally, all other factors being equal, the job with the highest queue priority is selected to be rolled in.

ERROR CONTROL

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

Errors may be detected by system software or hardware. When the system hardware detects an error condition, NOS issues two or more dayfile messages. The first message gives the address where the error was detected; the second and following messages give the types of errors that occurred. NOS then dumps the exchange package for the job either to OUTPUT, for batch and remote batch origin jobs, or to local mass storage file ZZZDUMP, for time-sharing jobs (refer to section 12). ZZZDUMP is not rewound before or after the dump.

After issuing the appropriate dayfile message(s) for the error(s), the system searches for an EXIT control statement. If an EXIT statement is found, processing continues with the statement following EXIT. If an EXIT statement is not encountered, the system terminates the job. (Exit processing is further described in section 5.) If the user issues a NOEXIT statement, the system does not search for an EXIT statement on subsequent errors, and processing continues with the next control statement.

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

Volume 2 describes the EREXIT, RECOVR, REPRIEVE, and MODE macros that can be used to control error processing in COMPASS programs. The SETLOF macro, described in volume 2, specifies file completion procedures when a job step abort occurs.

SECURITY CONTROL

A job cannot dump or directly change the contents of the job field length immediately after processing a protected control statement or user program. A COMPASS program can request protection through the SETSSM macro (refer to section 6, volume 2). These security restrictions do not apply if the job is of system origin or if the user is validated for system origin privileges and debug mode has been set at the system console.

A load/dump central memory utility control statement cannot immediately follow a protected control statement.

Protected Control Statements

ACCOUNT	COPY	DISPLAY	IFE	SKIP
ASSIGN	COPYBF	EDIT	LABEL	TCOPY
BEGIN	COPYBR	ELSE	LIBEDIT	USER
BLANK	COPYCF	ENDIF	REQUEST	VERIFY
CALL	COPYCR	ENDW	RESOURC	VFYLIB
CATALOG	COPYEI	ENQUIRE	RESTART	WHILE
CHARGE	COPYSBF	GOTO	REVERT	
CKP	COPYX	IF	SET	

Load/Dump Central Memory Control Statements

DMD	DMP	LBC	PBC	WBR
DMDECS	DMPECS	LOC	RBR	

If the user attempts to change or dump protected memory, NOS issues an informative message to the dayfile and aborts the job step.

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 released.
2. ECS assigned to the job is released.
3. All equipment assigned to the job is released.
4. All library files attached to the job are released; other jobs can then access them.
5. All scratch (local) file space used by the job is released.
6. All direct access permanent files attached to the job are released; the status information for these files is updated.
7. The following summation of the number of cards read through a local or remote batch reader is put in the beginning of the job dayfile. This information is also issued to the associated account dayfile for site usage. The entries in the account dayfile also include the job name.

- Job name:

  hh.mm.ss.jobname.

- Cards read in kilocards:

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

  mi Machine ID.

  es EST ordinal of the output device.

8. The following summations of job activity are added to the end of the job dayfile. This information is also issued to the associated account dayfile for site usage. The entries in the account dayfile also include the job name.

   - Adder activity in kilounits (incremented by USER statements, CHARGE statements, and resource assignments).

     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.

---

†If the installation defines a CPU multiplier value, the value given is the product of the actual CPU seconds and the multiplier. The installation may assign a CPU multiplier value to each CPU within a dual-processor machine (refer to the NOS System Maintenance Reference Manual).

The following information is printed at the end of all print file listings.

- Lines printed in kilolines:

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

  or

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

  mi Machine ID.

  es EST ordinal of the output device.

  The UCLV summation is issued if the V carriage control character was used (refer to appendix I).

The following information is issued to the account dayfile only.

- Cards punched in kilocards:

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

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

10. All deferred routed print and punch files are released to the print and punch queues. The files named OUTPUT, PUNCH, PUNCHB, and P8 are also released to the queues, unless the user discards job output (for example, using the N parameter in the SUBMIT control statement).

CDC CYBER CONTROL LANGUAGE 4

CYBER Control Language (CCL) is the set of control statements that determine the processing sequence within the control statement record. CCL statements can insert control statements from a procedure file and conditionally or unconditionally skip control statements. To determine the conditions for transfer of control, CCL can interrogate the system for error flags, file status, device type, and current subsystem. The following subsections describe the statement syntax and the operators and operands which make up a CCL expression. Following that is a discussion of CCL statements, their formats, and their use of expressions. The last subsection discusses CCL procedures which can contain CCL statements and expressions.

NOTE

Another system control language, KCL (described in appendix H), is also available. Support of KCL will be dropped in a future NOS release, so users are encouraged to convert their KCL procedures to CCL and not to mix the KCL and CCL statements. If the CCL BEGIN statement is mixed with the KCL GOTO or CALL statements, unpredictable results may occur.

STATEMENT SYNTAX

CCL statement syntax is similar to the syntax of all other control statements. The syntax rules are:

- A comma or left parenthesis separates the statement name and the first parameter.
- Commas separate consecutive parameters.
- A period or a right parenthesis terminates the statement.
- A right parenthesis ending an expression within a statement cannot also serve as the statement terminator. The user must include an additional right parenthesis or period to complete the statement.
- Parentheses can nest expressions within expressions (parentheses do not imply multiplication).
- Comments can follow the statement terminator.

Unlike most NOS control statements, a CCL statement can be longer than 80 characters. It can extend over more than one line if each line to be continued contains no more than 80 characters and ends with a separator.

OPERATORS

Operators separate operands in a CCL expression. There are three types of CCL operators: arithmetic, relational, and logical. Operators are used in the expressions within the IFE, WHILE, DISPLAY, and SET statements and the FILE function.

ARITHMETIC OPERATORS

Integer arithmetic is used in each step of the evaulation of a CCL expression. Division, multiplication, and exponentiation produce a zero result if the absolute value exceeds $2^{48} - 1$. Computations are accurate to 10 decimal digits (20 octal digits) and overflow is ignored.

The following are the CCL arithmetic operators.

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

RELATIONAL OPERATORS

A relational operator produces a value of 1 if the relationship is true, and 0 if it is false. The following are the CCL relational operators (either form may be used).

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

LOGICAL OPERATORS

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

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

ORDER OF EVALUATION

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

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

Operators of equal order are evaluated from left to right.

OPERANDS

One or more operands separated by operators make up a CCL expression. Expressions are used within the IFE, WHILE, DISPLAY, and SET statements. An expression within an expression must begin with a left parenthesis and end with a right parenthesis. There is no limit on the length of an expression, except that a period or a right parenthesis (not acting as a statement terminator) must appear within the first 50 operands. Expressions can contain operands of one or more types. There are three types of operands; constants, symbolic names, and functions.

CONSTANTS

A constant is a string of 1 to 10 characters that CCL processes as an integer. If its first character is a digit (0 through 9), all characters within the string must be digits, except the final character which may be a postradix. A B postradix identifies an octal integer; a D postradix identifies a decimal integer. If no postradix is specified, decimal is assumed.

If the first character of the constant is not a digit, it must be entered as a literal. A literal is a string of from 1 to 10 characters delimited by dollar signs (for example, $ A LITERAL $). CCL interprets the literal as right-justified display code with binary zero fill and processes it as an integer.

SYMBOLIC NAMES

A symbolic name is a string of characters that is recognized by CCL and has an assigned value. CCL uses symbolic names to test for conditions. It can also display the value assigned to a symbolic name.

The value assigned to a symbolic name is defined by the installation or set either by the user or by CCL. All variable symbolic names have an initial value of 0 except OT (job origin type), SYS (host operating system), VER (operating system version number), and TIME (the current time of day).

The symbolic names used with the FILE and DT functions are listed with the descriptions of the functions in this section. The following symbolic names can be used in CCL expressions. They are grouped according to a shared attribute.

- Symbolic names whose values are passed to, but not from, a procedure (refer to the description of procedures later in this section). When a procedure reverts, they are restored to the values they held when the procedure was called.

Name	Description
DSC	Flag determining whether skipped control statements are entered in the dayfile (refer to SET Statement in this section).
EF	Previous error flag.
R1	Control register 1 contents.
R2	Control register 2 contents.
R3	Control register 3 contents.

- Symbolic names whose values can be set by the user. All except EM are set by the SET control statement or the SETJCI macro (refer to section 6 of volume 2).

Name	Description
DSC	Flag determining whether skipped control statements are entered in the dayfile.
EF	Previous error flag.
EFG	Global error flag.
EM	Current exit mode (refer to MODE Statement, section 6).
R1	Control register 1 contents.
R1G	Global control register 1 contents.

Name	Description
R2	Control register 2 contents.
R3	Control register 3 contents.

- Symbolic names whose values are set by the operating system.

Name	Description
CMN	CM RFL setting divided by 100_8 (refer to RFL Statement, section 6).
DSC	Flag indicating that skipped control statements are to be entered in the dayfile.
ECN	ECS RFL setting divided by 100_8 (refer to RFL Statement, section 6).
EF	Previous error flag.
FL	Current CM field length.
MFL	Maximum CM field length.
MFLL	Maximum ECS field length.
OT	Job origin type.
SYS	Host operating system.
TIME	Current time of day (hhmm).
VER	Operating system version number.

- Symbolic name whose value is set by the calling or termination of a procedure.

Name	Description
PNL	Procedure nesting level (0 when processing the original control statement record, 1 when processing a first level procedure, and so forth). Its maximum value is 50.

- Symbolic name whose value can be set by the termination of a procedure (refer to SET Statement in this section).

Name	Description
EFG	Global error flag.

- Symbolic names correspond to error code values. In an expression a user typically checks the error flag (EF) for a nonzero value; a nonzero value indicates an error, and a zero value indicates no error. For detailed error examination, the user can compare EF with a particular symbolic name or its error code value. Users are encouraged to use the symbolic name, because the numeric values can change in future releases of NOS. The following list contains the errors that allow exit processing.

Name	Value	Description
ARE	1	Arithmetic error.
CPE	4	CPU abort.
ECE	15	ECS parity error.
FLE	7	File limit.
FSE	10	Forced error.
MNE	5	Monitor call error.
MSE	8	Mass storage error (same as track limit).
ODE	11	Operator drop.
OKE	13	Operator kill.
PPE	3	PPU abort.
PSE	2	Program stop.
RRE	12	Rerun error.
SRE	9	SRU limit.
SSE	14	Subsystem abort.
TKE	8	Track limit.
TLE	6	Time limit.

- Symbolic names with fixed values that can be compared with the origin type (OT) value within an expression.

Name	Description
BCO	Local batch origin.
EIO	Remote batch origin.
SYO	System origin.
TXO	Time-sharing origin.

- Symbolic name with a fixed value that can be compared with the host operating system (SYS) value within an expression.

Name	Description
NOS	Network Operating System.

- Symbolic names with true or false values. True is 1; false is 0.

Name	Description
F	Fixed value of 0 (false).
FALSE	Fixed value of 0 (false).
SWn	One of six sense switches (n can be from 1 to 6). Their values are set by the OFFSW, ONSW, and SWITCH statements (refer to section 6).
T	Fixed value of 1 (true).
TRUE	Fixed value of 1 (true).

FUNCTIONS

Functions are used as expressions or operands within expressions in CCL statements. Functions are not control statements. The CCL functions are FILE, DT, NUM, and SS.

FILE Function

The FILE function determines whether a file has a specified attribute. The system returns a value of true (1) or false (0) depending upon whether the file has or does not have the specified attribute(s). Only the equipment number (EQ) and file ID attributes can return values other than 1 or 0. The list of file attributes follows the description of the FILE function format.

The FILE function must be used as an expression or as a part of an expression in a CCL statement. A left parenthesis must appear before the file name, a comma must appear between the file name and the expression, and a right parenthesis must appear after the expression.

The format of the FILE function is:

FILE(lfn,exp)

lfn — Name of the file for which attributes are being determined.

exp — Either a special FILE function attribute or an expression, consisting of logical operators and special FILE function attributes. The expression must be appropriate for the statement in which the FILE function appears. If the FILE function is part of an IFE statement, the expression should be one that can be evaluated as true or false. If the FILE function is part of the DISPLAY statement, the expression could have a numeric value other than a true or false value.

The expression within a FILE function cannot include the NUM function, the SS function, or another FILE function; the DT function or the following symbolic names can be used within the expression. Any other symbolic name within the expression is treated either as an implicit DT function (refer to DT Function which follows) or an an unidentified variable.

Name	Attribute
AS	File is assigned or attached to the user's control point.
BOI	File is positioned at BOI. This is effective only for a file on mass storage.
EOF	Last operation was a forward operation, which encountered an EOF and is now positioned at that EOF. This is effective only for a file on mass storage.
EOI	Last operation was a forward operation, which encountered an EOI and is now positioned at that EOI. This is effective only for a file on mass storage.
EQ	EST number of the equipment on which the file resides. If the file is not assigned to the job, it has an equipment number of zero.
EX	File has execute-only permission.
ID	File ID value.
IN	File type is input.
LB	File is on a labeled tape.
LI	File type is library.
LO	File type is local.
MD	File has modify permission.
MS	File is on mass storage.
OP	File is opened.
PH	File type is punch.
PM	File is an attached direct access permanent file.
PR	File type is print.
PT	File type is primary.
RA	File has read append permission.
RD	File has read permission.
RM	File has read modify permission.
TP	File is on magnetic tape.
TT	File is assigned to a terminal.
WR	File has write permission.

Example:

The following job segment shows the FILE function being used inside an IFE control statement. The FILE function determines if file ACCT is not at the beginning-of-information (BOI). If ACCT is not at BOI, the IFE statement is true and the system rewinds ACCT before copying it onto ITEM. If ACCT is at BOI, the IFE statement is false and the system skips to the ENDIF control statement and copies ACCT onto ITEM. In both cases, ACCT is copied to ITEM and is replaced.

```
IFE,FILE(ACCT,.NOT.BOI),LABEL1.
REWIND,ACCT.
ENDIF,LABEL1.
COPY(ACCT,ITEM)
REPLACE,ITEM.
```

DT Function

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

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

FILE(lfn,DT(dt))

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

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

Type	Equipment
CP	415 Card Punch.
CR	405 Card Reader.
DE	Extended core storage.
DI	844-21 Disk Storage Subsystem (half-track).
DJ	844-41 or 844-44 Disk Storage Subsystem (half-track).
DK	844-21 Disk Storage Subsystem (full-track).
DL	844-41 or 844-44 Disk Storage Subsystem (full-track).
DM	885 Disk Storage Subsystem (half-track).
DP	Distributive data path to ECS.
DQ	885 Disk Storage Subsystem (full-track).

Type	Equipment
LP	Any line printer.
LR	580-12 Line Printer.
LS	580-16 Line Printer.
LT	580-20 Line Printer.
MT	Magnetic tape drive (seven-track).
NE	Null equipment.
NT	Magnetic tape drive (nine-track).
TT	Time-sharing terminal.

Example:

The following dayfile segment shows that TAXES is on a nine-track magnetic tape, so it is copied to output and then unloaded. If the DT function was false or if TAXES was not on magnetic tape, TAXES would be unloaded without being copied.

```
14.00.45.IFE,FILE(TAXES,TP.AND.DT(NT)),LABL1.
14.00.46.COPY,TAXES,OUTPUT.
14.00.46. EOI ENCOUNTERED.
14.00.46.ENDIF,LABL1.
14.00.46.UNLOAD,TAXES.
```

NUM Function

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

The format of the NUM function is:

NUM(c)

c A string of 1 to 40 characters. If the string contains one or more special characters, it must be delimited by dollar signs ($***$). If delimited by dollar signs, the string is always evaluated as nonnumeric.

Example:

The following procedure uses the NUM function to ensure that the passed parameter, NUMBER, is numeric. If a nonnumeric value is passed, the procedure terminates with an appropriate message.

```
.PROC,PROC1,NUMBER.
IFE,NUM(NUMBER),QUIT.
WHILE,R1.LE.NUMBER,LOOP.
SET,R1=R1+1.
          .
          .
          .
ENDW,LOOP.
REVERT. PROCESSING COMPLETED
ENDIF,QUIT.
REVERT,ABORT. NONNUMERIC PASSED
```

SS Function

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

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

The format of the SS function is:

SS

or

SS=name

name One of the following subsystem identifiers:

ACCESS	BATCH	FORTRAN	NULL
BASIC	EXECUTE	FTNTS	TRANACT†

The SS function is intended for use at a time-sharing terminal to determine and set subsystems by means of procedure calls. For example, a time-sharing user in the batch subsystem could call a procedure containing the statement SET,SS=FTNTS. Upon termination of the procedure, the user remains in the FTNTS subsystem.

CCL STATEMENTS

The following are the CCL control statements grouped according to their common functions.

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

Statement	Description
SKIP	Skips until a matching ENDIF statement is found.
IFE	Evaluates a conditional expression. If its expression is true, the next statement is processed; if its expression is false, statements are skipped until a matching ELSE or ENDIF statement is found.
ELSE	Terminates skipping initiated by a false expression within an IFE statement, or initiates skipping to a matching ENDIF statement.
ENDIF	Terminates skipping initiated by a matching IFE, SKIP, or ELSE statement.

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

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

† Not applicable to IAF.

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

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

The following CCL statements initiate and end processing of a procedure.

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

Individual descriptions of the control statements follow in alphabetic order.

BEGIN STATEMENT

The BEGIN statement inserts a procedure into the control statement record or into another procedure (refer to Procedures in this section). The procedure is stored in a procedure file. After the final control statement in the procedure is processed, a CCL- or user-supplied REVERT statement is executed and job processing continues with the statement following the BEGIN control statement. Use of a BEGIN statement is illustrated in figure 1-4-1.

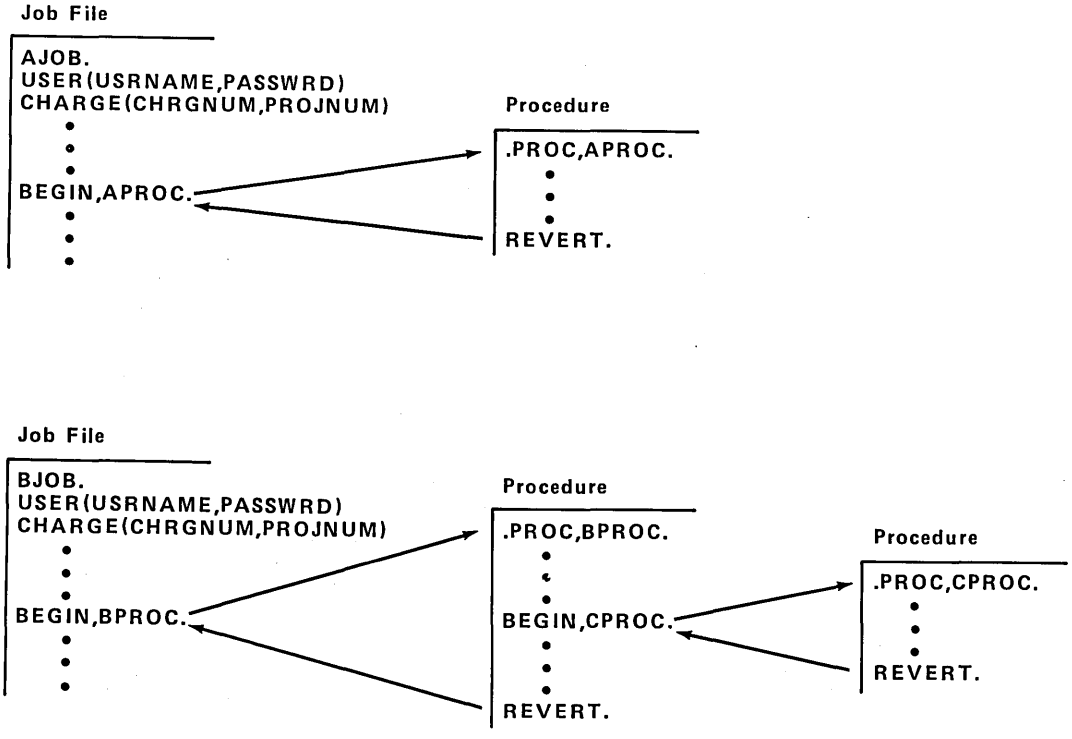


Figure 1-4-1. Calling a Procedure

The formats of the BEGIN statement are:

BEGIN,pname,pfile,p_1,p_2,...,p_n.

and

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

pname — Procedure name from the procedure header.

In the first format, pname is the name of a procedures on pfile.

If pname is omitted from the first format, two consecutive commas must be specified. The default procedure is the record at the current position of pfile. If pfile is at its end-of-information, CCL rewinds pfile and uses its first record. If pfile is INPUT, the file is not rewound.

In the second format, pname is the name of the local file containing the procedure pname or the name of a procedure on the system library. pname must be specified in the second format.

pfile — Name of the file containing the procedure. pfile must be the second parameter in the first format. Its omission is indicated by two consecutive commas following pname.

If pfile is omitted from the first format, the installation-defined default file name is used (PROCFIL is the default).

When the BEGIN statement is processed, CCL looks for a file named pfile assigned to the job. If none exists, it looks for an indirect access file named pfile and retrieves a local copy. If pfile is a direct access permanent file, the user must attach the file before the BEGIN statement is processed.

p_i — Optional parameter specifying the substitution to be made for a keyword used in the procedure. Refer to Keyword Substitution in this section for a full description of keyword use in procedures.

The following parameter formats are available.

keyword — The parameter is identical to a keyword on the procedure header, so the second default for the keyword is used (as specified on the procedure header).

keyword= — References to the keyword in the procedure are removed (null substitution).

value — CCL assigns this 1- to 40-character symbolic name or value to the keyword whose position in the procedure header parameter list matches the position of this parameter in the BEGIN statement parameter list. A value containing special characters, other than / or -, must be $-delimited.

keyword=value — The symbolic name or value is substituted for the keyword wherever it appears in the procedure. If value is followed by a +, value must be a symbolic name. (Refer to Symbolic Names earlier in this section.) Keyword in the BEGIN statement is the same keyword that is used in the procedure header statement.

The following formats can be used.

Format	Description
keyword=value or keyword=symbol	Substitutes the value or symbolic name itself.
keyword=symbol+ or keyword=symbol+D	Substitutes the decimal value associated with the symbolic name.
keyword=symbol+B	Substitutes the octal value associated with the symbolic name or interprets the symbolic name as an octal value.

When calling a procedure, a keyword can be named more than once if the keyword=value parameter format is used each time. CCL issues a message informing the user that a keyword is named more than once on the statement. It uses the value specified with the last occurrence of the keyword.

Example:

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

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

```
10.15.26.BEGIN,TEST1,FKTEST,20.
10.15.27.COMMENT.     20
10.15.27.REVERT.CCL
10.15.27.SET(R2=100)
10.15.27.BEGIN,TEST1,FKTEST,FK=R2+.
10.15.28.COMMENT.     100
10.15.28.REVERT.CCL
10.15.28.BEGIN,TEST1,FKTEST,FK=R2+D.
10.15.29.COMMENT.     100
10.15.29.REVERT.CCL
10.15.29.BEGIN,TEST1,FKTEST,FK=R2+B.
10.15.30.COMMENT.     144
10.15.30.REVERT.CCL
10.15.30.BEGIN,TEST1,FKTEST.
10.15.31.COMMENT.     FK
10.15.31.REVERT.CCL
10.15.31.BEGIN,TEST1,FKTEST,FK=.
10.15.32.COMMENT.
10.15.32.REVERT.CCL
10.15.32.BEGIN,TEST1,FKTEST,VALUE.
10.15.33.COMMENT.     VALUE
10.15.33.REVERT.CCL
10.15.33.BEGIN,TEST1,FKTEST,VALUE-2.
10.15.33.  CCL212- SEPARATOR INVALID    VALUE-
10.15.33. CPU ABORT.
10.15.33. JOB REPRIEVED.
10.15.33. CCL263- EXTERNAL ABORT DURING BEGIN
10.15.33.EXIT.
10.15.34.BEGIN,TEST1,FKTEST,$VALUE-2$.
10.15.34.COMMENT.     VALUE-2
10.15.34.REVERT.CCL
```

DISPLAY STATEMENT

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

The format of the DISPLAY statement is:

DISPLAY(exp)

exp A CCL expression.

Example:

The following sample dayfile shows several display operations.

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

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

ELSE STATEMENT

The ELSE statement performs one of the following functions.

- It terminates skipping initiated by a false IFE statement whose label string matches that of the ELSE statement. If the label string does not match, the ELSE statement is skipped.
- It initiates skipping from the ELSE statement to the ENDIF statement whose label string matches that of the ELSE statement. This happens for a true IFE statement.

Neither a SKIP nor an ELSE statement terminates skipping initiated by another SKIP or ELSE statement.

The format of the ELSE statement is:

ELSE(ls)

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

Example:

The following control statements use the FILE function to determine if a file named TEST1 is local to the job. If the file is local, it is copied to the OUTPUT file; if it is not, it is assumed to be an indirect access permanent file, and a local copy is obtained and copied to OUTPUT.

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

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

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

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

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

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

ENDIF STATEMENT

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

The format of the ENDIF statement is:

ENDIF(ls)

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

Example:

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

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

ENDW STATEMENT

The ENDW statement identifies the end of the WHILE control statement loop. A control statement loop is a sequence of statements that may be repeatedly processed. The number of times the loop is processed depends on the evaluation of the expression specified in the WHILE statement that begins the loop.

The ENDW statement must have a label string that matches the label string specified in the WHILE statement that begins the loop.

The format of the ENDW statement is:

ENDW(ls)

ls — Label string; 1 to 10 characters beginning with an alphabetic character. The string cannot contain special characters.

Refer to WHILE Statement in this section for an example of ENDW statement use.

IFE STATEMENT

The IFE statement conditionally initiates the skipping of succeeding statements. If the expression in the IFE statement is true, the next statement is processed. If the expression is false, CCL skips statements until it encounters a matching ELSE or ENDIF statement. The statements match when their label strings are identical.

An IFE statement must have a matching ELSE or ENDIF statement. If the IFE statement initiates skipping without a matching terminating statement, CCL aborts the job step. If the IFE statement is in a procedure, the terminating statement must also be in that procedure.

The format of the IFE statement is:

IFE,exp,ls.

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

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

Example 1:

The following control statements initiate the compilation and execution of a FORTRAN program and then test for any errors during execution. If an error was made, the error code is displayed.

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

If the job step executes without error, the error flag (EF) is 0. In this case, control passes to the ENDIF,LABEL1 statement. If an error occurs, the error flag is not 0, the statement is true, and control passes to the next statement; CCL then displays the error code in the error flag register. (The NOEXIT and ONEXIT statements are described in section 6.)

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

```
11.23.41.FTN,I=IFTEST.
11.23.44.       .017 CP SECONDS COMPILATION TIME
11.23.44.SET(EF=0) INITIALIZE ERROR FLAG
11.23.44.NOEXIT.
11.23.44.LGO.
11.23.45.      NON-FATAL LOADER ERRORS -
11.23.45.  UNSATISFIED EXTERNAL REF -- BETA
11.23.46. CPU ERROR EXIT AT  404253.
11.23.46. CM OUT OF RANGE.
11.23.48.ONEXIT.
11.23.48.IFE,EF.NE.0,LABL1.
11.23.48.DISPLAY(EF)
11.23.48.      1     1B
11.23.48.ENDIF,LABL1.
```

Example 2:

The following procedure file is an indirect access file called COLORPR. It uses the IFE statement to determine if the color the BEGIN statement substituted for COLOR is red or blue. Different processing is done for the colors red and blue. Any other color is ignored. The # character in the comment line inhibits substitution for the word (COLOR) it precedes (refer to Procedure Body later in this section).

```
.PROC,A,COLOR.
IFE,$COLOR$.EQ.$RED$,L1.
COMMENT.   PROCESSING DONE FOR #COLOR OF COLOR
REVERT.
ENDIF,L1.
IFE,$COLOR$.EQ.$BLUE$,L2.
COMMENT.   PROCESSING DONE FOR #COLOR OF COLOR
REVERT.
ENDIF,L2.
COMMENT.   NO PROCESSING FOR #COLOR OF COLOR
```

The following control statements call procedure A.

```
BEGIN,A,COLORPR,BLUE.
BEGIN,A,COLORPR,RED.
BEGIN,A,COLORPR,PINK.
```

The following dayfile segment results when the preceding control statements are processed. It shows the effect of the # character.

```
08.34.30.BEGIN,A,COLORPR,BLUE.
08.34.32.IFE,$BLUE$.EQ.$RED$,L1.
08.34.32.ENDIF,L1.
08.34.32.IFE,$BLUE$.EQ.$BLUE$,L2.
08.34.32.COMMENT.  PROCESSING DONE FOR COLOR OF BLUE
08.34.32.REVERT.
08.34.33.BEGIN,A,COLORPR,RED.
08.34.34.IFE,$RED$.EQ.$RED$,L1.
08.34.34.COMMENT.   PROCESSING DONE FOR  COLOR OF RED
08.34.34.REVERT.
08.34.34.BEGIN,A,COLORPR,PINK.
08.34.35.IFE,$PINK$.EQ.$RED$,L1.
08.34.35.ENDIF,L1.
08.34.35.IFE,$PINK$.EQ.$BLUE$,L2.
08.34.35.ENDIF,L2.
08.34.36.COMMENT.  NO PROCESSING FOR COLOR OF PINK
08.34.36.REVERT.CCL
```

REVERT STATEMENT

The REVERT statement terminates procedure processing. The formats are:

REVERT.comment

and

REVERT,ABORT.comment

comment — Optional character string appended after the statement terminator. This comment is especially useful to the time-sharing user because, when the REVERT statement is displayed at the terminal following procedure processing, the comment can inform the user as to how the procedure reverted. The REVERT statement is displayed at the terminal only if the time-sharing user is in the BATCH subsystem. The REVERT,ABORT statement is always displayed at the terminal.

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

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

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

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

Example:

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

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

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

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

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

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

```
10.09.51.REVJOB1.
10.09.51.USER(USERNUM,,FAMNAME)
10.09.51.CHARGE(CHARGNUM,PROJNUM)
10.09.51.ATTACH(FILE1/UN=ALTUSER,PW=,M=E)
10.09.52.BEGIN,REVTST,PROCFL,FILE1,XFIL.
10.09.53.IFE,FILE(FILE1,RD),LABEL1.
10.09.53.ELSE(LABEL1)
10.09.53.REVERT.NO READ PERMISSION
10.09.53.COMMENT. RETURNS HERE
10.09.54.EXIT.
```

```
10.10.11.REVJOB2.
10.10.11.USER(USERNUM,,FAMNAME)
10.10.11.CHARGE(CHARGNUM,PROJNUM)
10.10.11.ATTACH(FILE2/UN=ALTUSER,PW=,M=R)
10.10.12.BEGIN,REVTST,PROCFL,FILE2,XFIL.
10.10.14.IFE,FILE(FILE2,RD),LABEL1.
10.10.14.TDUMP(I=FILE2)
10.10.14. TDUMP COMPLETE.
10.10.14.ELSE(LABEL1)
10.10.14.ENDIF,LABEL1.
10.10.15.IFE,FILE(FILE2,RM),LABEL2.
10.10.15.ELSE(LABEL2)
10.10.16.REVERT,ABORT. NO READ/MODIFY PERMISSION
10.10.16.EXIT.
10.10.16.COMMENT. EXIT ON ERROR
```

SET STATEMENT

The SET statement assigns a value to a control register, an error flag, or the flag that determines whether skipped control statements are entered in the dayfile. Using the SS function, it also can change the current time-sharing subsystem.

To assign a value to a symbolic name, the following format is used.

SET(sym=exp)

sym — One of the following symbolic names (initially these names are set to 0).

Name	Description
R1, R2, or R3	Local control registers. When a procedure is called, the current values of R1, R2, and R3 are passed to the procedure. The values of these registers may change within the procedure. However, when processing reverts, these registers are restored to the values they had when the procedure was called.
R1G	Global control register. When a procedure is called or reverts, R1G keeps its current value.
EF	Local error flag. When a procedure is called, the current value of the error flag is passed to the procedure. The value of the error flag may change within the procedure. However, when processing reverts, the error flag is restored to the value it had when the procedure was called.
EFG	Global error flag. When a procedure is called or reverts, EFG keeps its current value.
DSC	Dayfile-skipped-control-statement flag. Initially, it is set to 0, so that control statements that are skipped (not processed) are not entered in the dayfile.

exp — A CCL expression. The value derived through evaluation of the expression is assigned to the symbolic name. Acceptable values for each symbolic name follow.

sym	Suggested Value
R1, R2, R3, or R1G	Any integer between -131 071 and 131 071. If the value is outside this range, it is truncated. CCL does not issue a message as a result of the truncation.

sym	Suggested Value
EF or EFG	Any integer between 0 and 63. If the value is greater than 63, it is truncated. To assign the value defined by the system for an error condition, the user should set the error flag to one of the error condition symbolic names (refer to Symbolic Names at the beginning of this section). CCL sets the EF flag to the appropriate error code when an error occurs. If EFG is 0 when a REVERT statement is processed, CCL sets EFG to the value in EF.
DSC	1 or 0. If the value of the expression is nonzero, DSC is set to 1. While DSC is 1, skipped control statements are entered in the dayfile preceded by two periods. Some CCL error processing routines set DSC to 1 so that skipped control statements are written in the dayfile.

To change the current time-sharing subsystem, the following format is used.

SET(SS=subsystem)

subsystem Subsystem name. The subsystem names are ACCESS, BASIC, BATCH, EXECUTE, FORTRAN, FTNTS, NULL, and TRANACT.†

Examples:

The first three examples use procedures from the following procedure file. It is an indirect access permanent file with the name SETFILE.

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

† TRANACT is not applicable to IAF.

Example 1 - Control Register Use:

The following control statements (below on the left side) set and display registers R1 and R1G. A procedure, P1, is called which displays these registers, resets them, and then reverts to the control statement record where they are again displayed.

On the right is the dayfile segment resulting from processing of the control statements.

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

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

Example 2 - Error Flag Use (EFG Nonzero):

The following control statements (below on left side) set values in the error flags EF and EFG and then call a procedure which attempts to access an indirect access permanent file. Control reverts to the control statement record where EF and EFG are displayed to see if any error code generated is returned via these flags. On the right side is the dayfile segment resulting from the processing of the control statements.

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

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

Example 3 - Error Flag Use (EFG Zero):

To return the error code generated in a procedure to the control statement record, EFG must be 0 before there is an exit from the procedure. This is demonstrated by the following control statements (below left side).

The dayfile segment (on the right) resulting from processing of the statements shows how the error code is returned.

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

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

Example 4 - DSC Flag Use:

The following control statements (below on left side) demonstrate the effect of DSC=0 and DSC=1. On the right side is the dayfile segment resulting from processing of the preceding control statements.

```
SET(DSC)=0)                              16.49.36.SET(DSC=0)
SKIP(LABL1)                              16.49.36.SKIP(LABL1)
COMMENT. SINCE THE DAYFILE SKIP          16.49.36.ENDIF(LABL1)
COMMENT. CONTROL IS SET TO ZERO,         16.49.37.SET(DSC=1)
COMMENT. THESE STATEMENTS WILL NOT       16.49.37.SKIP(LABL2)
COMMENT. APPEAR IN THE DAYFILE.          16.49.37...COMMENT. SINCE THE DAYFILE SKIP
ENDIF(LABL1)                             16.49.37...COMMENT. CONTROL IS NOW SET TO ONE,
SET(DSC=1)                               16.49.37...COMMENT. THESE STATEMENTS WILL
SKIP(LABL1)                              16.49.37...COMMENT. APPEAR IN THE DAYFILE AND
COMMENT. SINCE THE DAYFILE SKIP          16.49.37...COMMENT. EACH WILL BE FLAGGED
COMMENT. CONTROL IS NOW SET TO ONE,      16.49.37...COMMENT. WITH TWO INITIAL PERIODS.
COMMENT. THESE STATEMENTS WILL           16.49.37.ENDIF(LABL2)
COMMENT. APPEAR IN THE DAYFILE AND
COMMENT. EACH WILL BE FLAGGED
COMMENT. WITH TWO INITIAL PERIODS.
ENDIF(LABL2)
```

SKIP STATEMENT

The SKIP statement initiates unconditional skipping of succeeding control statements. Skipping is terminated by an ENDIF statement that has a label string matching the label string specified on the SKIP statement. Only an ENDIF statement, and not an ELSE statement, terminates skipping initiated by a SKIP statement.

The format of a SKIP statement is:

SKIP,ls.

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

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

WHILE STATEMENT

The CCL iterative statements WHILE and ENDW bracket a group of control statements into a loop that can be repeatedly processed. The beginning of the loop is identified by a WHILE statement and the end by an ENDW statement. The ENDW statement must have a label string that matches the label string specified on the WHILE statement. The loop is repeated as long as the expression in the WHILE statement is true. If the expression is initially false, control immediately skips to the ENDW statement; if no ENDW statement is found, all the remaining statements in the control statement record are skipped.

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

The format of the WHILE statement is:

WHILE,exp,ls.

exp	A CCL expression. The separator following exp must be a comma.
ls	Label string; 1 to 10 alphanumeric characters, beginning with an alphabetic character.

Example:

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

```
SET(R1=0)
SET(R2=5)
WHILE,R1.LT.R2,FINISH.
SET(R1=R1+1)
DISPLAY(R1)
    .
    .
    .
ENDW,FINISH.
```

The user can vary the number of repetitions by setting different values in R2.

PROCEDURES

A procedure is a group of control statements which exist apart from the job control statement record. A BEGIN statement inserts a procedure into the control statement record of a job or into another procedure previously called by the job. A procedure is to a control statement record as a subroutine is to a program. Like a subroutine, a procedure usually contains the control statements required to perform a single function within a job. A procedure can be changed by parameters passed to the procedure from the BEGIN statement.

A procedure is stored as a record on a file. Several procedures can exist on one file. The file may be a local file, an indirect access permanent file, or an attached direct access permanent file. A procedure file can reside on magnetic tape as well as on mass storage.

PROCEDURE STRUCTURE

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

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

NOTE

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

A CCL statement can extend over more than one line if each line to be continued contains no more than 80 characters and ends with a separator. If a new line is not a legal CCL statement or command, it is interpreted as a control statement error when executed.

Procedure Header Statement

The procedure header statement is the first line of the procedure. It identifies the procedure and specifies the keywords for the BEGIN statement. The BEGIN statement substitutes the keywords into the procedure body. Unless the header statement contains an error, it does not appear in the dayfile.

The syntax rules for header statements are:

- The header statement must begin with a period followed by the characters PROC.
- The separators between parameters must be commas.
- A period terminates the header statement.
- The header statement can extend over more than one line if each line to be continued ends with a separator.

The format of the procedure header statement is:

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

pname — Name of the procedure; one to seven alphanumeric characters. It can begin with a numeric character.

p_i — Optional formal parameters whose keywords are used in the body of the procedure. Depending on the BEGIN statement parameters, keywords in the procedure body can be removed, left as they are, replaced by a value specified in the BEGIN statement, or replaced by first or second default values as specified on the procedure header parameter (refer to Keyword Substitution in this section).

The maximum number of procedure header keywords is defined by the installation. The default is 50.

The following are the legal formats for p_i.

Format	Example
keyword	FILE1
keyword=	FILE1=
keyword=default1	FILE1=LGO
keyword=default1/default2	FILE1=LGO/OLD
keyword=/default2	FILE1=/OLD
keyword=#DATA (CDC graphics: keyword=≡DATA)	FILE1=#DATA
keyword=#FILE (CDC graphics: keyword=≡FILE)	FILE1=#FILE

keyword — A 1- to 10-character keyword.

default1 — A 1- to 40-character first default value. If default1 contains special characters, it must be $-delimited. This default value replaces the keyword if this parameter is omitted from the BEGIN statement.

default2 — A 1- to 40-character second default value. If default2 contains special characters, it must be $-delimited. This default value replaces the keyword if the BEGIN statement specifies a parameter value identical to the keyword.

default1 and default2 could be either of the following special values.

#DATA — Special default value used for keyword if an overriding value is not specified on the BEGIN statement. If this default value is used, the keyword within the procedure body references the record that immediately follows the procedure record on the file (refer to figure 1-4-2).

#FILE — Special default value used for keyword if an overriding value is not specified on the BEGIN statement. If this default value is used, the keyword within the procedure body references a data file created within the procedure by CCL procedure commands (refer to Procedure Commands in this section).

Procedure Body

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

Program input from record
on separate file, INFILE

Processing of the second procedure,B1,is initiated with the control statement:

BEGIN,B1,PROFIL1,INFILE.

Program input from next record
on file containing procedure

Processing of the second procedure,B2,is initiated with the control statement:

BEGIN,B2,PROFIL2.

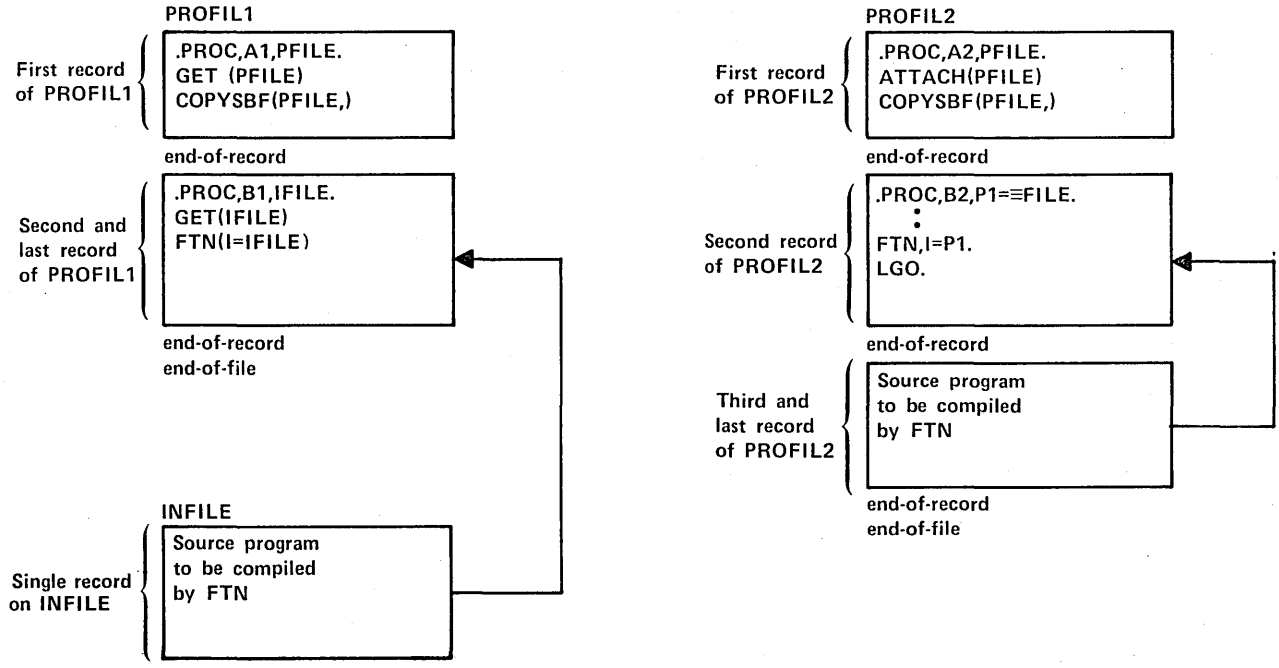


Figure 1-4-2. Procedure Access to a Data Record

When specifying keywords in the procedure body, two special characters, ASCII graphics # and _ (or CDC graphics ≡ and↱), are used to inhibit keyword substitution and to combine parts of a parameter after keyword substitution.

A single # character placed immediately before a keyword in a procedure statement inhibits substitution for that keyword. Two such characters (##) placed immediately before a keyword allow substitution; one # is retained. If # is placed before a nonkeyword, it has no effect; substitution takes place. If ## is placed before such a parameter, one # is retained. The # does not affect a separator.

The linking character, underline (_), is used in a procedure statement to temporarily separate two parameters (keyword or nonkeyword). After possible substitutions are made, the underline character is removed and the two parameters are merged into one. # before _ retains _ and allows substitution. _ before # does not affect the inhibiting action of #.

Examples of use of the # and _ characters in a procedure are shown in table 1-4-1.

TABLE 1-4-1. ALTERATIONS OF PARAMETERS IN A PROCEDURE BODY BY USE OF # AND _

Call statement: BEGIN,APROC,APROCFL.		
Procedure header: .PROC,APROC,FK1=X,FK2=Y.		
Procedure Parameters in Procedure Body before Substitution	Procedure Parameters in Procedure Body after Substitution	Comment
#FK1,FK1	FK1,X	# inhibits substitution in a keyword that immediately follows.
I,J	I,J	
FK1#FK2	XFK2	
I#J	IJ	
##FK1,FK1	#X,X	## allows substitution if a keyword immediately follows; one # is retained.
##I,J	#I,J	
##FK1=FK2	#X=Y	
FK1#,FK1	X,X	# does not affect a separator.
FK1_FK2	XY	_ separates two parameters before substitutions are made; after all substitutions are made, they are joined into one parameter.
I_J	IJ	
FK1_J	XJ	
I_FK2	IY	
FK1#_FK1	X_X	# before _ retains _ and allows substitution.
FK1#_FK2	X_Y	
FK1_#FK1	XFK1	_ before an # does not affect the inhibiting action of the #.

Example 1: # Character

The following procedure resides on file PROCFIL.

```
.PROC,INHIBIT,I=TEST.
GET(I)
FTN(#I=I,L=0)
LGO.
COMMENT.  I, #I, I#I, #I#I.
```

On the left is the BEGIN statement that calls procedure INHIBIT. On the right is the resulting dayfile.

```
BEGIN,INHIBIT,,.
```

```
14.14.24.$BEGIN,INHIBIT,,.
14.14.24.GET(TEST)
14.14.24.FTN(I=TEST,L=0)
14.14.27.       .022 CP SECONDS COMPILATION TIME
14.14.27.LGO.
14.14.28.      END  PROG1
14.14.28.    011500 MAXIMUM EXECUTION FL.
14.14.28.       .002 CP SECONDS EXECUTION TIME.
14.14.28.COMMENT.  TEST, I, TESTI, II.
14.14.28.$REVERT.CCL
```

Example 2: _ Character

Procedure LINK resides on file FILE1.

```
.PROC,LINK,TYPE=SBF,LFN1,LFN2.
REWIND(LFN1)
COPY_TYPE(LFN1,LFN2)
```

The first BEGIN statement does a COPYSBF of file PLAN to file SCHEME. The next BEGIN statement does a COPYEI of file MAZE to file TAXES. Each resulting dayfile follows the BEGIN statement.

```
BEGIN,LINK,FILE1,TYPE=SBF,LFN1=PLAN,LFN2=SCHEME.

 08.00.17.$BEGIN,LINK,FILE1,TYPE=SBF,LFN1=PLAN,LFN2=SCHEME.
 08.00.18.REWIND(PLAN)
 08.00.18.COPYSBF(PLAN,SCHEME)
 08.00.18. EOI ENCOUNTERED.
 08.00.18.$REVERT.CCL

BEGIN,LINK,FILE1,TYPE=EI,LFN1=MAZE,LFN2=TAXES.

 08.03.23.$BEGIN,LINK,FILE1,TYPE=EI,LFN1=MAZE,LFN2=TAXES.
 08.03.23.REWIND(MAZE)
 08.03.24.COPYEI(MAZE,TAXES)
 08.03.24. EOI ENCOUNTERED.
 08.03.24.$REVERT.CCL
```

PROCEDURE COMMANDS

Procedure commands enable the user to format a data file within a procedure and to insert documentary comments within a procedure. The commands are in fixed format with a period in column 1 and the command name beginning in column 2.

.DATA Command

A .DATA command in a procedure marks the beginning of a sequence of data lines to be written to a separate file when the procedure is called. File marks generated by .EOR and .EOF commands can subdivide the lines written to the data file. The sequence of data lines is terminated by one of the following:

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

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

The format of the .DATA command is:

.DATA,lfn

lfn — Optional name of the file on which the data lines are to be written. If a file named lfn is already assigned to the job, it is released, and new local file lfn is created. After the data file is written, it is automatically rewound.

If lfn is omitted, the default file referenced by special default #DATA is used. At the first procedure level, the system calls this file ZZCCLAA; at the second procedure level it is called ZZCCLAB; and so forth.

The following examples show three different ways of inserting a FORTRAN program into a procedure.

Example 1: Procedure accesses program data with .DATA command

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

```
.PROC,ALPHA,P1=#DATA,X=FTNOUT.
FTN(I=P1,L=X)
LGO.
REPLACE(X=LISTFIL)
.DATA.  FORTRAN PROGRAM FOLLOWS
      PROGRAM X(OUTPUT)
              .
              .
         FORTRAN SOURCE
            PROGRAM
              .
              .
      END
```

The following call statement in a control statement record of the job accesses procedure ALPHA on file DATAFIL.

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

A sample of a resulting dayfile is:

```
15.32.15.$BEGIN,ALPHA,DATAFIL.
15.32.16.FTN(I=ZZCCLAA,L=FTNOUT)
15.32.20.       .027 CP SECONDS COMPILATION TIME
15.32.20.LGO.
15.32.22.      END  FTNOUT
15.32.22.    011500 MAXIMUM EXECUTION FL.
15.32.22.       .004 CP SECONDS EXECUTION TIME.
15.32.22.REPLACE(FTNOUT=LISTFIL)
15.32.22.$REVERT.CCL
```

All input after the .DATA command (the FORTRAN source program) is written onto the default temporary file ZZCCLAA. Parameter substitution also occurs in the data statements.

Example 2: Procedure accesses program data with #FILE

The following procedure is an indirect access permanent file named PFILE. The record immediately following procedure BETA contains the program data. The #FILE default tells the FTN compiler to search for input from the next record on file BETA.

```
.PROC,BETA,P1=#FILE,X=FTNOUT.
FTN(I=P1,L=X)
LGO.
REPLACE(X=LISTFIL)
-EOR-
      PROGRAM X(OUTPUT)
              .
              .
         FORTRAN SOURCE
            PROGRAM
              .
              .
      END
```

The following call accesses procedure BETA on file PFILE.

```
BEGIN,BETA,PFILE.
```

The following is a segment of the resulting dayfile. Parameter substitution occurred within the procedure but not within the FORTRAN program.

```
10.15.33.$BEGIN,BETA,PFILE.
10.15.33.FTN(I=PFILE,L=FTNOUT)
10.15.37.       .026 CP SECONDS COMPILATION TIME
10.15.37.LGO.
10.15.39.      END  X
10.15.39.     011500 MAXIMUM EXECUTION FL.
10.15.39.       .005 CP SECONDS EXECUTION TIME.
10.15.39.REPLACE(FTNOUT=LISTFIL)
10.15.39.$REVERT.CCL
```

Example 3: Procedure accesses program data from another file

To access program data outside of the procedure file, the procedure must include a GET or an ATTACH control statement. The following procedure is in the default file PROCFIL. It uses a GET statement to access the program data on file TEST.

```
.PROC,GAMMA,P1=PROGRM,X=FTNOUT.
GET(P1)
FTN(I=P1,L=X)
LGO.
REPLACE(X=LISTFIL)
```

The following call accesses procedure file GAMMA.

```
BEGIN,GAMMA,,P1=TEST.
```

Parameter substitution occurred within the procedure but not within the FORTRAN program, as shown in the following dayfile segment.

```
09.15.40.$BEGIN,GAMMA,,P1=TEST.
09.15.40.GET(TEST)
09.15.40.FTN(I=TEST,L=FTNOUT)
09.15.44.       .030 CP SECONDS COMPILATION TIME
09.15.44.LGO.
09.15.46.      END  X
09.15.46.     011500 MAXIMUM EXECUTION FL.
09.15.46.       .004 CP SECONDS EXECUTION TIME.
09.15.46.REPLACE(FTNOUT=LISTFIL)
09.15.46.$REVERT.CCL
```

An example of a data file written from a procedure to a named file is shown in figure 1-4-3.

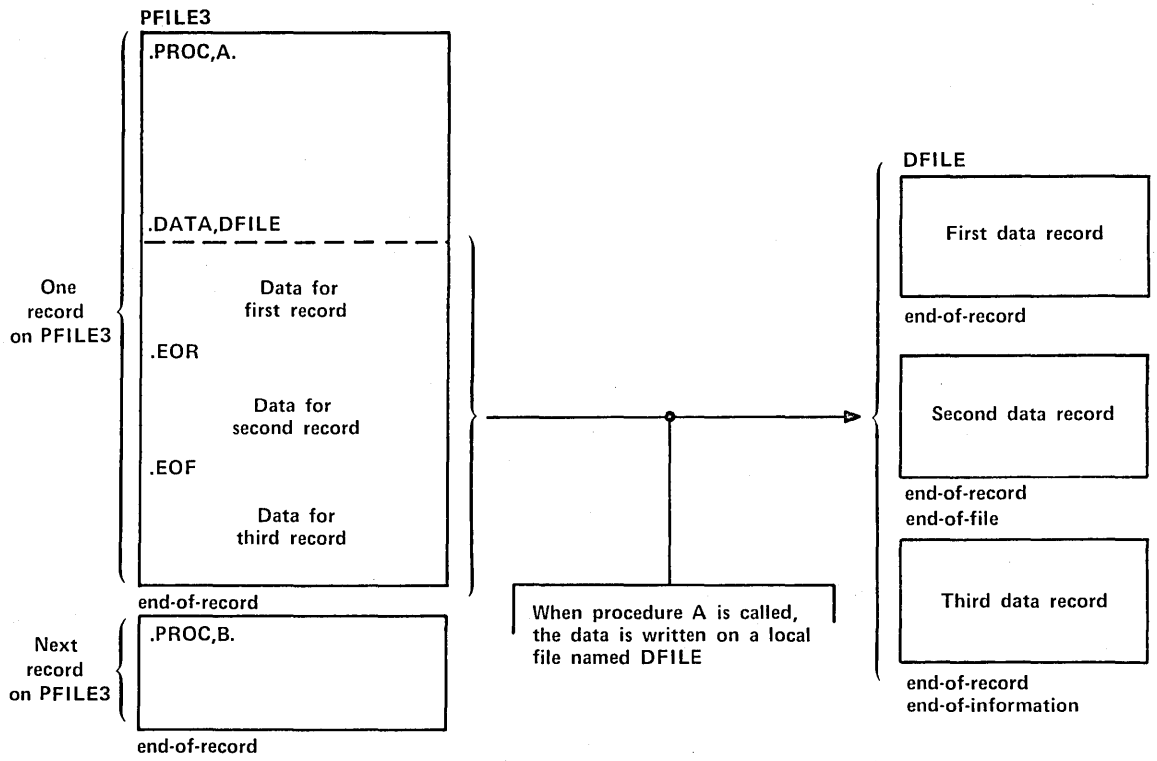


Figure 1-4-3. Data File Written from a Procedure to a Named File

.EOR Command

The .EOR command is used to separate records in a data file originating in a procedure. Whenever an .EOR is placed, an actual end-of-record is recorded when the data file is written on #DATA or lfn. Since the data statements are written on an external file, the .EOR command has no effect on the system end-of-record that terminates the procedure. The .EOR command is valid only after a .DATA command (refer to figure 1-4-3). A terminator must not be used and nothing else can appear on the same line.

.EOF Command

The .EOF command generates an end-of-file on the data file originating in a procedure. An actual end-of-file is recorded when the data statements are written on #DATA or lfn. This command has no effect on the end-of-record that terminates the procedure. If the end of the data file format is also the end of the procedure, no .EOF command is needed. In this case, an end-of-record mark is added. If the user wants an end-of-file mark, he must include an .EOF command. The .EOF command is valid only after a .DATA command (refer to figure 1-4-3). A terminator must not be used, and nothing else can appear on the same line.

.* Command

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

PARAMETER SUBSTITUTION

The user who creates a procedure uses parameters in the procedure header statement, which the BEGIN statement may change when the procedure is called. CCL can remove the parameter (null substitution), leave it as is, or replace it with a value from the procedure call statement or with a default value from the procedure header. The parameters are then substituted into the procedure body.

After substitutions are made in the procedure body, some control statements may be expanded beyond 80 characters. For most control statements, this is flagged as an error. Exceptions are CCL statements and the ASSIGN, BLANK, LABEL, REQUEST, and VSN statements, which can extend over more than one line if the statement is split at a separator. The user should ensure that the line containing the parameter is short enough so that possible expansion does not extend the line beyond the 80th character.

When a procedure is called, CCL must match each parameter on the call statement to a parameter on the header statement. CCL uses two methods of parameter matching, order dependent and order independent.

Order-Dependent Parameter Matching Mode

Parameter matching always begins in order-dependent mode (refer to Order-Independent Parameter Matching Mode for information on changing parameter matching modes). CCL compares, in order, each parameter on the BEGIN statement with the parameter in that position on the procedure header statement. CCL then substitutes the selected parameters into the procedure body.

All possible parameter substitutions in order-dependent mode are summarized in table 1-4-2. The table shows each parameter format on the BEGIN statement, each parameter format on the procedure header statement, and the substitution resulting from each combination. In the table the word value indicates that the parameter in the BEGIN statement (called value) is different than the corresponding keyword and/or defaults on the procedure header statement. Keyword in the BEGIN Statement Parameter Format heading means the keyword in the BEGIN statement is identical to the keyword in the procedure header statement parameter.

Assuming that all parameter matches between the BEGIN statement and the procedure header are valid for order-dependent mode (table 1-4-2), CCL completes parameter matching in order-dependent mode.

In order-dependent mode, CCL ignores excess parameters on the BEGIN statement.

The user should use table 1-4-2 with the following examples to clarify the meaning of the table entries (keyword, default1, default2, value, and null).

Examples: Parameter Matching in Order-Dependent Mode.

Procedure on File Named MYFILE	Call and Substitution	Explanation
.PROC,SAMPL1,L,M,N=XY. REWIND,L,A,M,N.	BEGIN,SAMPL1,MYFILE. yields REWIND,L,A,M,XY.	When parameters are omitted on the BEGIN statement, the system uses the defaults from the procedure header (L, M, and XY).
	BEGIN,SAMPL1,MYFILE,,,N. yields REWIND,L,A,M,N.	Omitted parameters indicate use of the procedure header defaults (L and M). N overrides the procedure header default (XY).
	BEGIN,SAMPL1,MYFILE,$*$,C. yields REWIND,*,A,C,XY.	Special characters must be $-delimited. The asterisk (*) replaces M. The system uses the procedure header default (XY) for the omitted parameter.
.PROC,SAMPL2,LFN1=,LFN2, SBF=/SBF. COPY_SBF(LFN1,LFN2)	BEGIN,SAMPL2,MYFILE. yields COPY(,LFN2)	Omitted parameters indicate use of procedure header defaults (LFN2 and null substitution for LFN1 and SBF).
	BEGIN,SAMPL2,MYFILE,,,SBF. yields COPYSBF(,LFN2)	Omitted parameters indicate use of procedure header defaults (null and LFN2). The BEGIN statement parameter, SBF, indicates use of the second default of the SBF procedure header parameter (SBF). The linking character (_) connects COPY and SBF to make COPYSBF.
	BEGIN,SAMPL2,MYFILE,FORMS, yields OUTPUT. COPY(FORMS,OUTPUT)	FORMS replaces LFN1= and OUTPUT replaces LFN2. Since the third parameter is omitted, the system uses the procedure header default (null).
.PROC,SAMPL3,PFN,P1=/$M=W$. ATTACH,PFN/P1.	BEGIN,SAMPL3,MYFILE,TAXES,P1. yields ATTACH,TAXES/M=W.	TAXES replaces PFN and the character string M=W replaces P1.

TABLE 1-4-2. PARAMETER SUBSTITUTION IN ORDER-DEPENDENT MODE

		BEGIN Statement Parameter Format				
		omitted	keyword	$keyword$	value	$value$
Procedure Header Parameter Format	keyword	keyword	keyword	keyword	value	value
	keyword=	null	keyword	keyword	value	value
	keyword=default1	default1	keyword	keyword	value	value
	keyword=default1/default2†	default1	default2	default2	error	error

†Switches keyword substitution to order-independent mode for all subsequent parameters.

Example 1:

The following procedure is on file PROCFIL. It prepares a file for processing. If the file is local, it is rewound. If it is not local, the system searches for the file in the user's permanent file catalog. If the file is not found, the procedure reverts and aborts.

```
.PROC,PREPARE,FNAME=,M=R.
IFE,FILE(FNAME,AS),PREP1.
REWIND(FNAME)
REVERT. FNAME PREPARED.
ENDIF,PREP1.
ATTACH(FNAME/#M=M,NA)
IFE,FILE(FNAME,.NOT.AS),PREP2.
GET(FNAME/NA)
IFE,FILE(FNAME,.NOT.AS),PREP3.
REVERT,ABORT. FNAME NOT FOUND.
ENDIF,PREP3.
ENDIF,PREP2.
REVERT. FNAME PREPARED.
EXIT.
REVERT,ABORT. PREPARE ERRORS.
```

The user prepares file TEST with the following statement.

```
BEGIN,PREPARE,,TEST.
```

Since PROCFIL is the default file, it does not have to be specified and is noted by successive commas.

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

```
08.26.45.$BEGIN,PREPARE,,TEST.
08.26.46.IFE,FILE(TEST,AS),PREP1.
08.26.46.ENDIF,PREP1.
08.26.46.ATTACH(TEST/M=R,NA)
08.26.46. TEST IS INDIRECT ACCESS,  AT 000121.
08.26.46.IFE,FILE(TEST,.NOT.AS),PREP2.
08.26.46.GET(TEST/NA)
08.26.47.IFE,FILE(TEST,.NOT.AS),PREP3.
08.26.47.ENDIF,PREP3.
08.26.47.ENDIF,PREP2.
08.26.47.REVERT. TEST PREPARED.
```

Example 2: Parameter Matching in Nested Procedures (Order-Dependent Parameter Matching Mode)

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

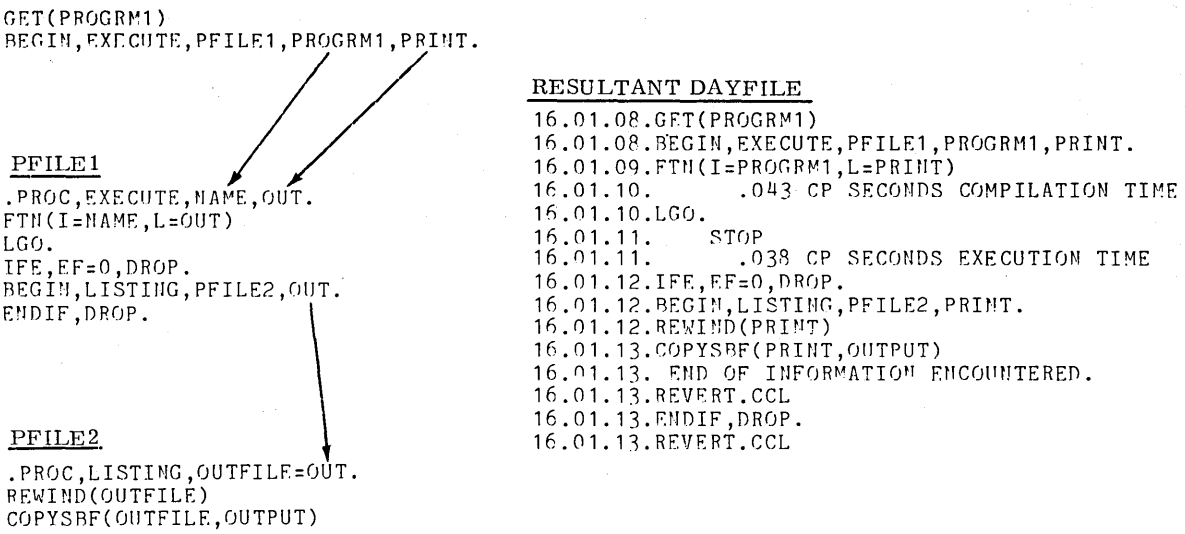


Figure 1-4-4. Keyword Substitution in Two Procedures

Order-Independent Parameter Matching Mode

CCL switches to order-indpendent mode to match the remainder of the parameters if in comparison of a BEGIN statement parameter and a procedure header parameter one of the following occurs.

- A BEGIN statement parameter is in the format keyword= or keyword=value.
- A procedure header statement parameter is in the format keyword=default/default2.

For each BEGIN statement parameter, matching always begins in order-dependent mode. Once in order-independent mode, CCL matches each keyword of the BEGIN statement to the identical keyword in the procedure header statement, regardless of order.

The following statements illustrate the parameter combinations that result in switching from order-dependent mode to order-independent mode.

Procedure on Default File PROCFIL	Call and Substitution	Explanation
.PROC,SALES,TAX,TOTAL=,FLAG=A. COPYL(TAX,TOTAL,HOLD,,FLAG) REPLACE(HOLD=TAX)	BEGIN,SALES,,TAX,TOTAL=SUM,FLAG. yields COPYL(TAX,SUM,HOLD,,A) REPLACE(HOLD=TAX)	Parameter matching starts in order-dependent mode. The BEGIN parameter TOTAL=SUM switches the mode to order-independent mode. FLAG is then matched in order-independent mode, which yields A.
.PROC,TAXES,TAX=FED/MN,DEDUCT, FLAG=A. COPYL(TAX,DEDUCT,HOLD,,FLAG) REPLACE(HOLD=TAX)	BEGIN,TAXES,,TAX,DEDUCT. yields COPYL(MN,DEDUCT,HOLD,,A) REPLACE(HOLD=MN)	The TAX=FED/MN procedure parameter switches the mode to order-independent mode. All parameters will be matched in order-independent mode.

All possible parameter substitutions in order-independent mode are summarized in table 1-4-3. The table shows each parameter format on the BEGIN statement, each parameter format on the procedure header statement, and the substitution resulting from each combination.

In the table the word value indicates that the parameter in the BEGIN statement (called value) is different than the keyword and/or defaults on the procedure header statement. The user should use table 1-4-3 with the following examples to clarify the meaning of the table entries (keyword, default1, default2, value, and null).

Examples of Parameter Matching:

Procedure on Default File PROCFIL	Call and Substitution	Explanation
.PROC,SAMPL1,L,M,N=XY. REWIND,L,A,M,N.	BEGIN,SAMPL1,,L=SWITCH. yields REWIND,SWITCH,A,M,XY.	The L=SWITCH BEGIN parameter switches parameter matching mode to order-independent mode. Order-independent mode uses the procedure header defaults for omitted BEGIN parameters. Order-dependent and order-independent modes work identically for omitted BEGIN parameters.
	BEGIN,SAMPL1,,L=CHANG,M,N. or BEGIN,SAMPL1,,L=CHANG,N,M. yields REWIND,CHANG,A,M,XY.	The L=CHANG parameter switches parameter matching to order-independent mode. In order-independent mode the order of the BEGIN parameters does not matter. M matches with M, and the N BEGIN keyword indicates substitution of the procedure header default (XY).
	BEGIN,SAMPL1,,L=FLIP,M=B,N=Z. yields REWIND,FLIP,A,B,Z.	BEGIN parameters in the form keyword=value always override procedure header parameters. FLIP replaces L, B replaces M, and Z replaces XY.
.PROC,TRACE,IN,OUT, TC=EOI,N=1. COPY(IN,OUT,,,TC,N)	BEGIN,TRACE,,IN=,OUT=HOLD,N=. yields COPY(,HOLD,,,EOI,)	The IN= parameter switches parameter matching to order-independent mode. All BEGIN statements in the form keyword= use null substitution.
.PROC,SAMPL4,FILE1, EC=B6/A6,DC=LP,REP=0. REWIND(FILE1) ROUTE(FILE1,#DC=DC, #EC=EC,#REP=REP)	BEGIN,SAMPL4,,COIN. yields REWIND(COIN) ROUTE(COIN,DC=LP,EC=B6,REP=0)	COIN is substituted in order-dependent mode. EC=B6/A6 switches the mode to order-independent mode. The omitted parameters indicate use of the procedure header defaults (order-dependent and order-independent modes work alike for omitted BEGIN parameters).
	BEGIN,SAMPL4,,COIN,EC,DC,REP. yields REWIND(COIN) ROUTE(COIN,DC=LP,EC=A6,REP=0)	EC=B6/A6 switches the mode to order-independent mode. Specifying the keyword on the BEGIN statement produces the same result as omitting them (refer to previous example) except for the double default procedure parameter, EC=B6/A6. If EC is omitted, B6 is used. If EC is specified, A6 is used.

TABLE 1-4-3. PARAMETER SUBSTITUTION IN ORDER-INDEPENDENT MODE

		BEGIN Statement Parameter Format				
		omitted	keyword or $keyword$	keyword= or $keyword$=	keyword=value or $keyword$=value	value or $value$†
Procedure Header Parameter Format	keyword	keyword	keyword	null	value	error
	keyword=	null	null	null	value	error
	keyword=default1	default1	default1	null	value	error
	keyword=default1/ default2	default1	default2	null	value	error

†Assumes the parameter is entered under order-independent mode.

Example 1:

The following procedure is the same procedure as Example 2 in Order-Dependent Parameter Matching Mode. It resides on file PROCFIL. It routes a specified file (FNAME) to the specified equipment (default is any CDC-graphics line printer).

```
.PROC,PRINTR,FNAME,REP=0,DC=LP,EC=B6.
REWIND(FNAME)
ROUTE(FNAME,#DC=DC,#REP=REP,#EC=EC)
REVERT.  FNAME ROUTED.
EXIT.
REVERT,ABORT.  PRINTR PARAMETER ERRORS.
```

The following control statement calls the procedure PRINTR. The system matches COLOR in order-dependent form. DC=SB switches the mode to order-independent mode. SB indicates the file is to be punched.

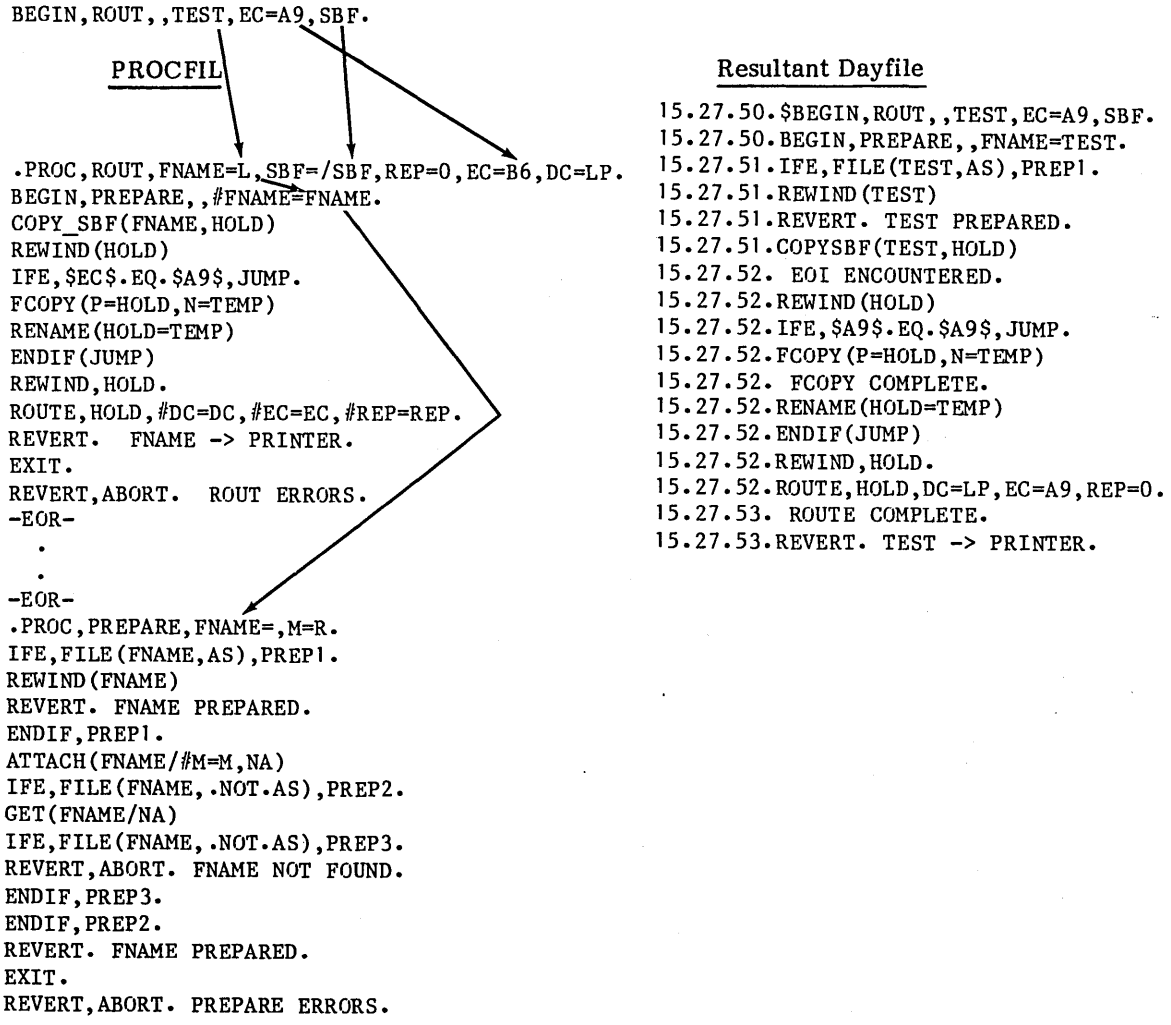
```
BEGIN,PRINTR,,COLOR,DC=SB,EC=SB.
```

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

```
15.27.26.$BEGIN,PRINTR,,COLOR,DC=SB,EC=SB.
15.27.27.REWIND(COLOR)
15.27.27.ROUTE(COLOR,DC=SB,REP=0,EC=SB)
15.27.27. ROUTE COMPLETE.
15.27.28.REVERT.  COLOR ROUTED.
```

Example 2: Parameter Matching in Nested Procedures (Order-Dependent and Order-Independent Parameter Matching Modes)

As shown in figure 1-4-5, procedures ROUT and PREPARE reside on the default file PROCFIL. A BEGIN statement within ROUT calls PREPARE. In procedure ROUT the substitution for the FNAME parameter (TEST) is passed to procedure PREPARE by the BEGIN statement. The resulting dayfile is on the right side of figure 1-4-5.



BEGIN calls procedure ROUT. The SBF=/SBF parameter switches parameter matching to order-independent mode. The first control statement of ROUT is a BEGIN statement that calls procedure PREPARE. The parameters are matched in order-independent mode. PREPARE readies a file for processing. If the file is local, it is rewound. If it is not local, the system searches for the file in the user's permanent file catalog. If the file is not found, the procedure reverts and aborts. If the file is found, processing continues with the second control statement in procedure ROUT. The file is prepared for printing. Since the file is to be printed with the ASCII graphic 95-character set (EC=A9), the file must be changed to a 12-bit ASCII code file (FCOPY). The procedure then routes the file to the printer and reverts to the statement following the BEGIN control statement.

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

CONTROL STATEMENT PROCESSING 5

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

The operating system recognizes three types of control statements.

- Local File Control Statements — These statements call files that are assigned to the job control point. LGO is the system default local file used for retaining object code generated by one of the language processors.
- System Control Statements — These statements are divided into eight categories.

  Job control control statements

  File management control statements

  Permanent file control statements

  Load and dump central memory utility control statements

  Tape management control statements

  System utility control statements

  Library utility control statements

  Loader control statements†
- Product Set Control Statements — The product set control statements call the various products available under NOS. Their formats are given in the applicable product reference manual and in the Applications Programmer's Instant.

CONTROL STATEMENT FORMAT

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

†Refer to the CYBER Loader Reference Manual.

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

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

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

The system allows continuation lines for CCL statements and ASSIGN, BLANK, LABEL, REQUEST, and VSN control statements (for details, refer to Statement Syntax in section 4 and Control Statement Rules in section 10).

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

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

Operating System Format	Product Set Format
4. All embedded blanks within a control statement except those appearing in literals are ignored.	4. Same as for the operating system format.
5. Comments may appear on the control statement but they must follow the terminator. They may contain any character. Comments are not printed for some control statements.	5. Same as for the operating system format.
6. Parameters, separators, and terminators are stored in the user's field length beginning at RA+2. The characters , . and) are stored as binary zero. For all parameters and all valid separators except the comma, their display code equivalent is stored. Refer to section 10 of volume 2 for more information.	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

Refer to section 10 of volume 2 for more information.

7. File names are one to seven alphanumeric characters.	7. File names are one to seven alphanumeric characters. In some products, file names beginning with a numeric character are illegal.
8. Not NOS/BE compatible.	8. NOS/BE compatible.

In general, no parameter can contain more than seven characters. If a parameter contains more than seven 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 seven characters. If a parameter contains more than seven characters, the ARG= entry point is not present, and the SDM= entry point is present (refer to appendix F in volume 2), the statement name (such as DEFINE) is issued to the dayfile but all parameters are suppressed.

Depending on the program, 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. A keyword is an identifier which has meaning either by itself or when used in conjunction with an option. Usually, keywords are passed with an option and a separator. The separator must not be a comma. When the list of parameters is passed to the called program, all separators except commas are also passed.

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

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

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

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

The parameters and an image of the control statement being processed are written in the job communication area (refer to section 10 of volume 2). The job communication area is the first 110_8 words of the user's field length, from RA through RA+107_8. Section 1 and appendix E in volume 2 describe the first 100_8 words of this area.

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

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

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

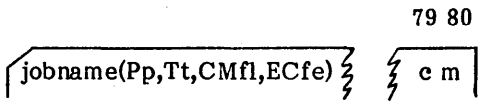
JOB STATEMENT (JOB CARD)

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

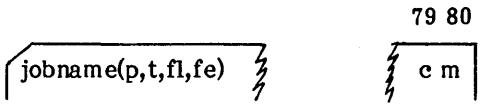
† Not applicable to time-sharing jobs.

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

The job statement format is:



or



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

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

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

CMfl or fl — Maximum octal field length for the job (refer to Field Length Control in section 3). The system rounds the value to the next highest multiple of 100_8. The default field length is the maximum field length for which the user is validated (refer to the LIMITS statement in section 6) or the maximum field length available for user jobs (dependent on machine size), whichever is smaller. The field length cannot exceed:

377700_8 on a 198 K or a 262 K machine

360000_8 on a 131 K machine

163000_8 on a 65 K machine

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

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

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

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

Example:

```
JOBAAA,,1,50000,20.
```

has the same effect as:

```
JOBAAA,T1,CM50000,EC20.
```

CONTROL STATEMENT PROCESSING FLOW

The system translates a control statement by:

1. Reading the statement from the job control statement buffer.
2. Verifying the format of the statement as described in Control Statement Format.
3. Comparing special control statement names with the name of the control statement being processed. If the statement name is CTIME, HTIME, RTIME, or STIME, the system processes the control statement.
4. Searching the file name table for a file assigned to the job with a name identical to the name of the control statement. However, if a $ precedes the program name, this step is skipped. If an identical name is found, the program is loaded into memory. The arguments are extracted from the control statement and stored in RA+2 through RA+n+1 (n is the number of parameters). The CPU is requested to begin execution unless special loader control statements follow.
5. Searching the central library directory for a program name that matches the control statement name. If the name is found, the system proceeds as in step 4; otherwise, the system searches further.

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

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

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

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

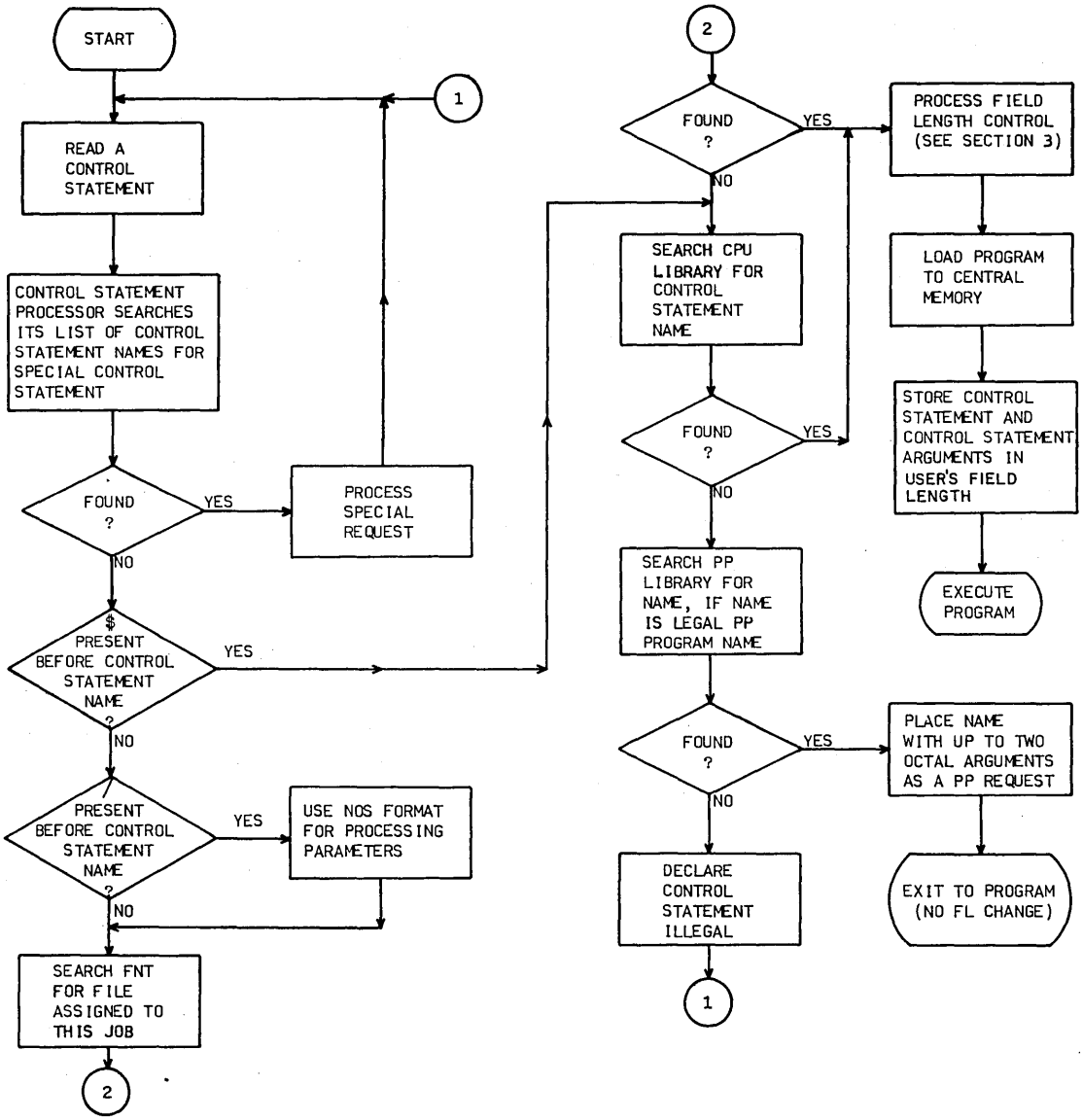


Figure 1-5-1. Control Statement Processing Flow

EXIT PROCESSING

When an error condition occurs during job processing, the system searches the control statement record for an EXIT statement. If the record does not contain an EXIT statement, the system terminates the job. If the system finds an EXIT statement, it clears the error condition and processes the control statements that follow the EXIT statement. If the error was a time limit error, the limit is reset to the time used plus 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 8 SRUs.

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

The following sequence of control statements illustrates this exit processing.

```
JOBCCC.
USER(SMITH22,SMA1)
CHARGE(55A19,69P5)
NOEXIT.
GET(A,B)
ONEXIT.
ATTACH(MASTER/M=W)
SKIPEI(MASTER)
COPYBF(A,MASTER)
COPYBF(B,MASTER)
PACK(MASTER)
COPYSBF(MASTER,)
EXIT.
ENQUIRE(F)
-EOR-
-EOI-
```

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

JOB CONTROL CONTROL STATEMENTS 6

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

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

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

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

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

ACCOUNT STATEMENT

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

CHARGE STATEMENT

The CHARGE statement causes the system to record on the account dayfile all information regarding resources used in the previous account block, and designates a new charge and project number for subsequent activity. Its purpose is to control the accounting activity of the system for a customer or the installation. An account block is that portion of a job from one CHARGE statement to the end of the job or to another CHARGE statement.

The control statement format is:

CHARGE(chargenum, projectnum)

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

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

chargenum,projectnum

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

If access option 7 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 7 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.

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

COMMENT STATEMENT

The COMMENT statement enters the specified comment in the system and user's dayfile. A COMMENT statement must not occur between the job and USER statements.

The control statement format is:

COMMENT.comments

or

*comments

comments Any combination of characters the user wishes to display.

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

CTIME STATEMENT

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

The control statement format is:

CTIME.

DAYFILE STATEMENT

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

The control statement format is:

DAYFILE(lfn,string,op,pd,pl,infile)

or

DAYFILE(L=lfn,FR=string,OP=op,PD=pd,PL=pl,I=infile)

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

FR=string Specifies a character string for which a search is to be made in the dayfile. The op parameter specifies the field to be searched.

The string specified must begin with the first character in the field to be searched. The time field begins with a blank.

If the string contains characters other than letters and numbers (such as blanks), it must be enclosed by $ delimiters. A $ within the string must be entered twice ($$) so it is not interpreted as a delimiter. Time-sharing commands entered in the dayfile are preceded by a $. To search for a time-sharing command, the $ preceding the command is replaced with two $'s, and the command is enclosed by $ delimiters (for example, $$$OLD$).

If the string is found, the portion of the dayfile following the last occurrence of the specified string is output. If the string is not found, an informative message and the entire dayfile is output.

OP=op Selects search option (single character):

op	Meaning
T	Search time field for string specified by FR=string.
M	Search message field for string specified by FR=string.
I	Incremental dump (dayfile printed from point of last dayfile statement).
F	Full dump.

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

PD=pd Print density (three, four, six, or eight lines per inch); if PD=pd is omitted, PD=6 is assumed.

PL=pl Selects page size; if PL=pl is 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

I=infile A copy of a dayfile is to be used for input. If I=infile is omitted, the DAYFILE statement uses the active dayfile for input.

NOTE

A paginated dayfile listing cannot be used as the input file (I=infile). Refer to L=lfn.

Examples:

DAYFILE,,$ 11.21.$,T.

The dayfile is dumped to OUTPUT starting at the last occurrence of 11.21. in the time field.

DAYFILE,FR=$$$RFL$.

The dayfile is dumped to OUTPUT starting at the last occurrence of $RFL in the message field. (The $RFL message is entered in the dayfile when a time-sharing user requests the batch subsystem.)

DAYFILE,I=DAY,FR=$$$GET,STATS.$

The dayfile is dumped to OUTPUT starting at the last occurrence of $GET,STATS. in the message field of the previously written dayfile named DAY.

The system may place diagnostic messages in the output produced by DAYFILE. They are not part of the user dayfile from which the DAYFILE output is produced. These messages begin with NOTICE*** and are described in appendix B.

ENQUIRE STATEMENT

The ENQUIRE control statement gives the user information about the job as it is being processed.

The control statement formats are:

ENQUIRE(OP=$p_1p_2...p_7$,JN=jobname,FN=lfn_1,O=lfn_2)

or

ENQUIRE($p_1p_2...p_7$)

If no parameters (other than O=lfn_2) are specified on an ENQUIRE statement in a batch job, the system assumes OP=A. In a time-sharing job, an ENQUIRE statement with no parameters outputs job status information (refer to the Network Products IAF Reference Manual or the NOS Time-Sharing User's Reference Manual).

OP=$p_1p_2...p_7$ — Any combination of the following options. The user can request a maximum of seven options on a statement (for example, ENQUIRE, OP=BDFJLRS.).

p_i	Description
A	Gives listings of the B, D, R, U, J, L, and F options.
B	Returns identification and priority information to the user.

Example:

```
SYSTEM ACTIVITY.

  USER NUMBER       SAH3333
  USER INDEX HASH   AOUY
  JOB NAME          AOUYCIE
  JOB SEQ.NO.       ACIE
  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 (refer to RESOURC Statement in section 6).

Example:

```
RESOURCE DEMAND INFORMATION.

 RESOURCE    DEMAND    ASSIGNED

   PE          1          0
   HD          1          0
```

p_i	Description
F	Gives the status of files at the user's control point. An asterisk (*) after the file type indicates that the file is locked. (The user cannot write on a locked file.) Refer to the FILE function in section 4 for the meaning of the file type mnemonics. The Status column lists the last operation performed on the file. (I/C means incomplete.)

Example:

```
LOCAL FILE INFORMATION.

 FILENAME  LENGTH/PRUS  TYPE  STATUS

  FIL1            1     PM.*  EOR READ
  AFIL            4     LO.   EOR WRITE
  INPUT           2     IN.*  EOR READ
  OUTPUT          3     PR.   I/C WRITE

 TOTAL = 10
```

J — Returns the contents of the user's control registers, error flag field, and succeeding control statements.

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

Example:

```
JOB CONTROL REGISTERS.

  R1 =   1
  R2 =   0
  R3 =   0
  EF =   0
 EFG =   0
 R1G =   0

CONTROL STATEMENT(S).

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

L — Returns user's loader information (refer to the CYBER Loader Reference Manual).

Example:

```
LOADER INFORMATION.
   MAP OPTIONS  =  DEFAULT
   GLOBAL LIBRARY SET IS -
LIB3
```

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

Example:

```
RESOURCES USED.

  CPU TIME              0.107 SECS.
  MS ACTIVITY           0.914 KUNS.
  MT ACTIVITY           0.000 KUNS.
  PF ACTIVITY           0.034 KUNS.
  ADDER                 0.003 KUNS.
  SRU                   3.157 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          3.162 UNTS.
```

T — Returns accumulated CPU time.

Example:

```
CPU ACCUMULATOR.

 CPU TIME      0.111 SECS.
```

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

Example:

```
RESOURCE USAGE ALLOWED.

  SECONDS            64
  JOB STEP SRU       31808
  ACCOUNT BLK SRU    31808
  DAYFILE MESSAGES   993
  CONTROL STATMTS    NO LIMIT
  DISPOSE FILES      12
  MASS STORAGE       59008
```

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

FN=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.

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

If the JN=jobname or FN=lfn_1 is executed, the information is printed on the OUTPUT file only if the OUTPUT file is assigned to an interactive terminal; otherwise, this information is written in the user's dayfile.

ENTER STATEMENT

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

The control statement format is:

ENTER./statement$_1$/statement$_2$/.../statement$_n$

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

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

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

Example:

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

```
batch
$RFL,0.
/enter.#get,fprog#ftn,i=fprog#map,part#lgo#dmp#rewind,zzzdump#copy,zzzdump
```

This is the sequence of control statements used in section 12 to illustrate the reading of CM dumps. Assuming that a FORTRAN Extended program is on the indirect access file FPROG, output like that shown in the figures in section 12 is produced when this statement is processed.

EXIT STATEMENT

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

The control statement format is:

EXIT.

HTIME STATEMENT

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

The control statement format is:

HTIME.

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

HTIME nnnnnnnnnnnn.nnn KCYCLES.

An HTIME statement processed on a machine other than the CYBER 170 Model 176 produces the following dayfile message.

HTIME NOT AVAILABLE.

LDI STATEMENT

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

The control statement format is:

LDI(lfn,id,m)

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

id — Identification code (0 through 67_8 and 77_8); if omitted, 0 is assumed. If an id of 77_8 is assigned, the special output files named OUTPUT, PUNCH, PUNCHB, and P8 are discarded at job completion, unless they have been explicitly routed.

m — Names of jobs loaded are listed in the dayfile for the submitting job; if omitted, the list is suppressed.

The number of executing jobs and output files a user has within the system cannot exceed the user's deferred batch job validation limit (refer to the DB field description for the LIMITS control statement in this section). If this limit is reached, no further jobs can be loaded until an existing job completes. The following message is issued to the dayfile.

TOO MANY DEFERRED BATCH JOBS.

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

ILLEGAL USER CARD.
SYSTEM ABORT.

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

SIUN,usernum.

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

LENGTH STATEMENT

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

The control statement format is:

LENGTH(lfn)

lfn Name of a file assigned to the job.

The information given for the local file includes its length in PRUs, type, and current status. Similar information can be obtained with the ENQUIRE control statement.

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 in seconds (cumulative CPU time slices) allowed for each job step of the user's job.
CM	Maximum number of central memory words that the user is allowed to request. The value stored for CM represents the actual word limit divided by 100_8.
NF	Maximum number of files that the user is allowed to have assigned to a job concurrently.
DB	Maximum number of deferred batch jobs that the user can have in the system concurrently. If the user is validated for system privileges and debug mode is set on the system display console or if the user is submitting jobs from system origin, this parameter is ignored. The user is allowed to submit as many jobs as desired.
FC	Maximum number of permanent files the user can have in the catalog.
CS	Maximum number of PRUs available to the user for indirect access files.
FS	Maximum number of PRUs available to the user for any one indirect access file.
PA†,††	Terminal parity (EVEN or ODD).
RO†,††	Number of rubout characters required for carriage return delay.
PX†,††	FULL or HALF duplex transmission mode.
TT†,††	Terminal type.
TC†	Character set to be used by time-sharing terminal.
IS†	Initial subsystem for time-sharing terminal.
MS	Maximum number of mass storage PRUs the user is allowed to additionally allocate via his job.
DF	Maximum number of CPU program messages that the user's job can issue to the system and/or job dayfiles.

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

††These fields are not used with network terminals.

Field	Description
CC	Maximum number of batch control statements processed for a user. Time-sharing control statements (listed in appendix E) are excluded.
OF	Maximum number of print and punch files the user can dispose to output queues.
CP	Maximum number of cards that can be punched from a user's punch file.
LP	Maximum number of lines that can be printed from a user's print file.
EC	Maximum number of ECS memory words that the user is allowed to request divided by 1000_8.
SL	Maximum number of SRUs the user is allowed for a job.
CN	Charge number to which the user is assigned.
PN	Project number to which the user is assigned.
DS	Maximum number of PRUs available to the user for any one direct access permanent file.
AW	Access word; controls the user's access within the system according to the following options (assumed values are options 0, 2, and 3).

Option	Specifies
0	User can change his password.
1	User can use the privileged time-sharing commands.†
2	User is allowed to create direct access files.
3	User is allowed to create indirect access files.
4	User can have system origin privileges for any job origin if the system console is in debug mode. The user is allowed to assign a device by its EST ordinal although the system need not be in debug mode to do so. The user is allowed to call the customer engineering PP-based diagnostics if engineering mode (ENGR) is set at the system console.
5	User can access/create library files.
6	User can assign magnetic tape units. Refer to the REQUEST statement in section 7 for further information.

†For further information about privileged time-sharing commands, refer to the NOS Time-Sharing User's Reference Manual or the Network Products IAF Reference Manual.

Field	Description	
	Option	Specifies
	7	User is allowed to access the system without supplying his assigned charge and project numbers.
	8	User can define, save, and replace files on auxiliary devices.
	9	User can access special transaction functions for library updates and batch transaction processing.
	10	Allows no terminal timeout.
	11	Allows use of the system control point (SCP) facility.
	12	User has special accounting privileges.†
	13	Allows BATCHIO subsystem privileges.††
	14	Allows use of the PROTECT statement.
	15-23	Reserved for Control Data.
	24-35	Used by Control Data for application validation.†††
	36-47	Available for user application validation.
	48-59	Reserved for installation.

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

AW=0000000000100000215

the user would be validated for options 0, 2, 3, 7, and 24, as shown in the following:

```
AW=        1  0  0  0  0  0  2  1  5
           ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~
           001000000000000000010001101

Options=   24                  7   32 0
```

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

ERROR IN LIMITS ARGUMENTS.

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

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

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

MFL STATEMENT

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

MFL(nnnnnn,mmmm)

or

MFL(CM=nnnnnn,EC=mmmm)

nnnnnn — Maximum central memory field length (octal is assumed unless decimal is specified by a D suffix or use of the digits 8 or 9).

mmmm — Maximum extended core storage (ECS) field length. The value of mmmm is the actual extended core field length divided by 1000_8.

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

The parameter nnnnnn sets an upper boundary for the field length of subsequent job steps. The value cannot exceed the maximum field length for the job nor can it be less than 1500, the field length required for the utility (CONTROL) that processes MFL. If ECS is assigned for the job, the CM field length cannot be less than 10 000. Likewise, the parameter mmmm sets an upper boundary for the ECS field length of subsequent job steps and cannot exceed the maximum field length for the job. If the value 0 (zero) is entered for CM or ECS field length, the MFL is set to either the maximum field length for which the user is validated or the field length specified on the job statement, whichever is smaller.

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

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

CM OR EC REQUEST EXCEEDS MAXIMUM.

MODE STATEMENT

The MODE statement allows the user to define the error conditions that cause the system to exit from normal processing. When the error that the user specified 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 user did not select the exit mode processing, the system ignores the error and continues normal processing.

The control statement format is:

MODE(m,n)

m — CPU program error exit mode ($0 \leq m \leq 17_8$). Modes 10_8 through 17_8 are only legal for the CYBER 170 Model 176. Selects the error condition(s) for which normal error processing does and does not occur. If no mode is selected, the default is m=7.

n — Included for compatibility with earlier versions of NOS. The system now ignores the value specified on the control statement.

The following values can be supplied for m.

m	Normal Error Processing Occurs	Error Ignored and Job Continues
0	None.†	Address out of range, operand out of range, indefinite operand, or underflow.††
1	Address out of range.	Indefinite operand, operand out of range, or underflow.††
2	Operand out of range.†	Address out of range, indefinite operand, or underflow.††
3	Address or operand out of range.	Indefinite operand or underflow.††
4	Indefinite operand.†	Address or operand out of range, or underflow.††
5	Address out of range or indefinite operand.	Operand or operand out of range, or underflow.††
6	Indefinite operand or operand out of range.†	Address out of range or underflow.††
7	Indefinite operand, address out of range, or operand out of range (default value).	Underflow.††
10,11†††	Underflow mode or address out of range.	Operand out of range, or indefinite operand.
12,13†††	Underflow mode, address out of range, or operand out of range.	Indefinite operand.
14,15†††	Underflow mode, address out of range, or indefinite operand.	Operand out of range.
16,17†††	Underflow mode, address out of range, operand out of range, or indefinite operand.	None.

Address out-of-range error is caused by an attempt to reference CM or ECS outside of established limits, or by an attempt to reference the last 60-bit word (word 7) in the relative address FL of ECS. Operand out-of-range error is caused by a floating-point arithmetic unit receiving an infinite operand. An indefinite operand error is caused by a floating-point arithmetic unit receiving an indefinite operand. Dividing zero by zero results in an indefinite operand. Underflow mode error is caused by a floating-point arithmetic unit receiving a negative exponent value that is too small. For further information about the processing of error mode errors, refer to Error Control in section 3 and to the CYBER 170, CYBER 70, and 6000 Series Computer Systems reference manuals.

---

† On the CYBER 170 Model 176, address out of range (m=1) is always selected.
†† Underflow is applicable only to the CYBER 170 Model 176.
††† These modes are legal only on the CYBER 170 Model 176. Since address out of range (m=1) is always selected, the two modes are equivalent.

NOEXIT STATEMENT

The NOEXIT control statement suppresses EXIT statement processing. If an error occurs, control is not transferred to the statement following the next EXIT statement. Instead, processing continues with the next control statement (unless the error causes the job to unconditionally terminate). Refer to the description of exit processing in section 5 for more information.

The control statement format is:

NOEXIT.

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 a data base when the job would otherwise be rerun by the operator.

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

NOTE STATEMENT

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

The control statement format is:

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

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

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

Example:

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

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

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

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

OFFSW STATEMENT

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

The control statement format is:

OFFSW(s_1,s_2,...,s_n)

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

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

ONEXIT STATEMENT

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

The control statement format is:

ONEXIT.

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

ONSW STATEMENT

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

The control statement format is:

ONSW(s_1,s_2,...,s_n)

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

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

PASSWOR STATEMENT

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

The control statement format is:

PASSWOR(oldpswd,newpswd)

oldpswd	Old password
newpswd	New password

The new password must be the minimum length required by the installation. The default minimum is four characters; its maximum length is seven characters. Only alphabetic and numeric characters can be used in the password.

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

oldpswd,newpswd

The user's password is changed from oldpswd to newpswd. The user can change his password only if access option 0 is set (refer to the LIMITS control statement in this section). If option 0 is not set and the user submits a PASSWOR statement, the system issues the following message to his dayfile.

ILLEGAL CONTROL CARD.

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

ERROR IN PASSWOR ARGUMENTS.

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

MODVAL ABORTED.

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

PROTECT STATEMENT

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

The control statement format is:

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

or

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

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

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

The PROTECT statement is available to the user only if option 14 of his access word is set (refer to the LIMITS control statement in this section). If option 14 is not set and the user submits a PROTECT statement, the system issues the following message to his dayfile.

CPM - ILLEGAL USER ACCESS.

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

ERROR IN CONTROL ARGUMENTS.

RERUN STATEMENT

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

The control statement format is:

RERUN.

If the RERUN statement has been issued, the operator can rerun the job if necessary. This statement is ignored from a time-sharing origin job.

RESOURC STATEMENT

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

The control statement format is:

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

rt_i Resource type:

LO	200 bpi, seven-track magnetic tape unit
HI	556 bpi, seven-track magnetic tape unit
HY	800 bpi, seven-track magnetic tape unit
HD	800 cpi, nine-track magnetic tape unit
PE	1600 cpi, nine-track magnetic tape unit
GE	6250 cpi, nine-track magnetic tape unit
MT†	Seven-track magnetic tape unit
NT†	Nine-track magnetic tape unit (800/1600 cpi)
DIi	844-21 Disk Storage Subsystem (1≤i≤8)
DJi	844-41/44 Disk Storage Subsystem (1≤i≤8)
DKi	844-21 Disk Storage Subsystem (full-track)(1≤i≤8)
DLi	844-41/44 Disk Storage Subsystem (full-track)(1≤i≤8)
DMi	885 Disk Storage Subsystem (half-track)(1≤i≤3)
DQi	885 Disk Storage Subsystem (full-track)(1≤i≤3)

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

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

DEADLOCK PREVENTION

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

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

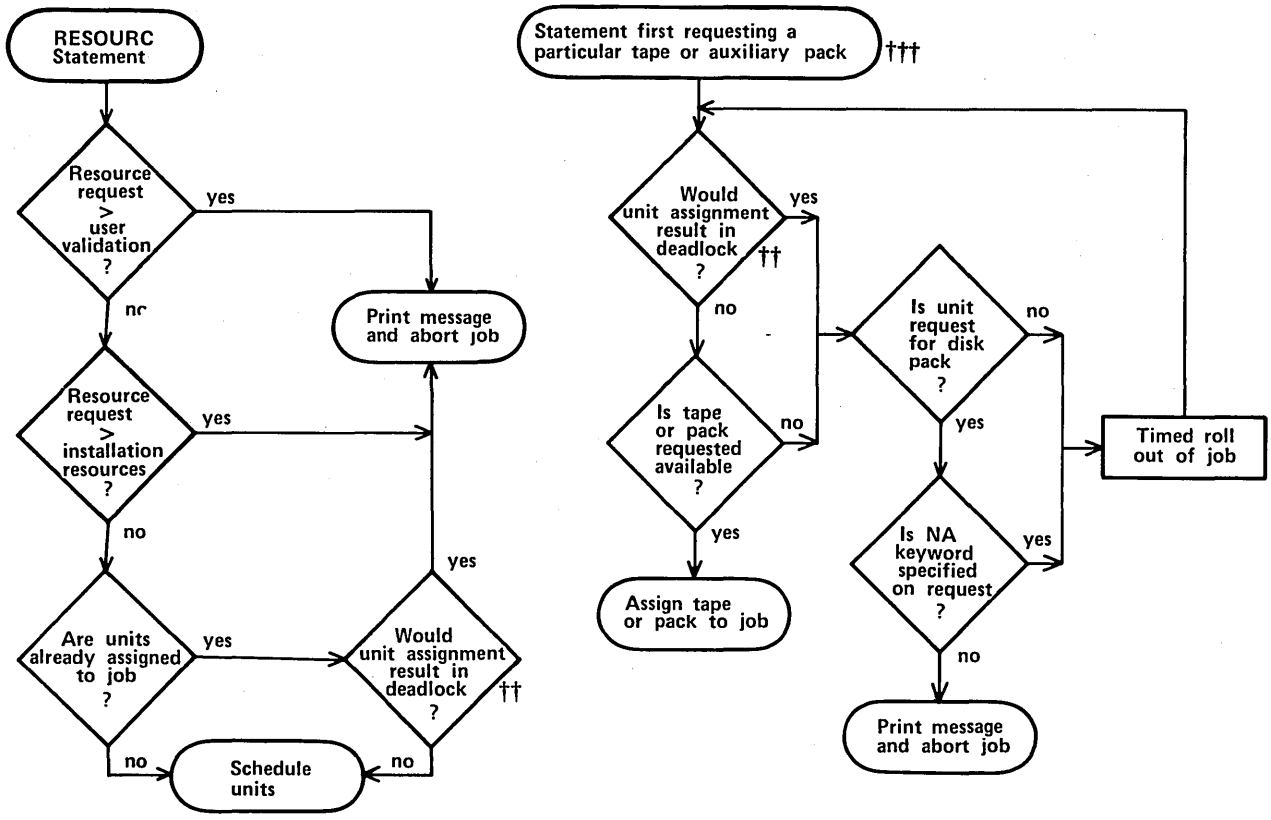


Figure 1-6-1. Resource Commitment Processing (Simplified)

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

††Refer to Resource Overcommitment later in this section.

†††The statements are described in sections 8 and 10.

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

SINGLE RESOURCE USE

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

TAPE UNITS

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

Examples:

Assume that an installation has the following tape drive resources:

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

---

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

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

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

   RESOURC(NT=3)

   the job is aborted with the message

   INSUFFICIENT RESOURCES ON SYSTEM.

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

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

   RESOURC(NT=1,PE=1)

   the job is aborted with the message

   CONFLICTING RESOURCE TYPES.

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

3. If a job contains the following control statements

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

   the job is aborted with the message

   CONFLICTING RESOURCE TYPES.

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

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

RESOURCE OVERCOMMITMENT

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

NOTE

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

ALTERING RESOURCE REQUIREMENTS

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

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

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

NOTE

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

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

Example:

The first RESOURC statement schedules two 800-cpi, nine-track tape units. The two LABEL statements assign the tape units to the job. Because the maximum scheduled units were used concurrently, the RETURN statement decreases the scheduled tape units to zero. The second RESOURC statement schedules two 844-21 disk units and one 800-cpi, nine-track tape unit.

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

UNIT ASSIGNMENT

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

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

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

NOTE

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

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

RFL STATEMENT

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

The control statement format is:

RFL(nnnnnn,mmmm)

or

RFL(CM=nnnnnn,EC=mmmm)

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

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

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

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

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

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

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

CM OR EC REQUEST EXCEEDS MAXIMUM.

ROLLOUT STATEMENT

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

The control statement format is:

ROLLOUT(t)

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

RTIME STATEMENT

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

The control statement format is:

RTIME.

The dayfile message format is:

RTIME nnnnnn.nnn SECS.

SETASL STATEMENT

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

The control statement format is:

SETASL(s)

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

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

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

SETCORE STATEMENT

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

The control statement format is:

SETCORE(p)

or

SETCORE(-p)

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

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

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

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

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

SETJSL STATEMENT

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

The control statement format is:

SETJSL(s)

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

s must be greater than 0 and less than or equal to the current account block SRU limit and the user's and the account's SRU validation limits. Exceptions to this rule are the asterisk (*) and values greater than 32 760 (77770B). These values set the job step SRU limit at the validation limit if the account block SRU limit is set at the validation limit.†

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

SETPR STATEMENT

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

SETPR(p)

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

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

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

SETTL STATEMENT

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

The control statement format is:

SETTL(t)

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

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

The time limit set is the next multiple of 8 greater than or equal to t. For example, SETTL(20) sets a time limit of 24.

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

To set a time limit for job step execution on one CPU of a dual-processor machine, the user must include a USECPU statement in the job. Otherwise, the time limit is set for the cumulative job step execution time on both CPUs.

STIME STATEMENT

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

The control statement format is:

STIME.

The dayfile message format is:

STIME nnnnnn.nnn UNTS.

SUBMIT STATEMENT

The SUBMIT control statement places a user-supplied job file into the input queue as a separate job. SUBMIT can reformat the file according to directives within the file. Refer to Job Names in section 3 for a description of the submitted job's job name.

The control statement format is:

SUBMIT(lfn,q,NR)c

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

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

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

NR — No rewind option; the submit file and the file specified on a cREAD reformatting directive are not rewound before or after processing. If NR is omitted, the files are rewound before, but not after processing.

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

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

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

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

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

ln1, ln2, and ln3 are optional line numbers.

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

The reformatting directives available are described as follows:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

NO READ FILE - lfn.

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

READ FILE BUSY - lfn.

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

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

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

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

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

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

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

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

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

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

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

```
SIUN,usernum.
```

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

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

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

EXIT.

DAYFILE(lfn)

REPLACE(lfn)

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=$p_1p_2...p_n$,JN=jobname,FN=lfn_1,O=lfn_2)

or

SUMMARY($p_1p_2...p_n$)

or

SUMMARY.

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

SWITCH STATEMENT

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

The control statement format is:

SWITCH($s_1,s_2,...,s_n$)

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

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

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

USECPU STATEMENT

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

The control statement format is:

USECPU(n)

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

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

USER STATEMENT

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

The control statement format is:

USER(usernum,passwrd,familyname)

usernum	A one- to seven-character alphanumeric user number.
passwrd	Alphanumeric password. Its maximum length is seven characters; its minimum length is defined by the installation.
familyname	Optional parameter identifying the family† of permanent file devices on which the user's permanent files reside. The user specifies a family name when the system can access more than one permanent file device family.

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

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

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

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

```
SIUN,usernum.
```

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

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

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

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

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

Normally, the familyname parameter need not be included on the USER statement. However, if the user makes a practice of specifying his family name each time he submits a job, he can be sure that his job will be processed even if his normal system is not available and his permanent file family is moved to a backup system. If, after the first USER statement, the user does not specify a familyname on the USER statement, his permanent file family remains the same. If the user specifies the 0 (zero) familyname, his permanent file family becomes the system default family.

Example:

An installation has two systems, A and B. System B provides backup service for system A. The system default family name for system A is AFAM, and the system default family name for system B is BFAM.

During normal operations, system A user CWJONES with password JPWD could enter either of the following USER statements.

```
USER(CWJONES,JPWD)

USER(CWJONES,JPWD,AFAM)
```

System B user JDSMITH with password SPWD could enter either of the following statements.

```
USER(JDSMITH,SPWD)

USER(JDSMITH,SPWD,BFAM)
```

If system A failed, user CWJONES would be required to enter

USER(CWJONES,JPWD,AFAM)

to identify his family of permanent file devices. User JDSMITH could enter either of the USER statements as before because the default family name would still be valid.

If the user attempts to access permanent files on a device not present in the alternate system, one of the following messages is issued to the user's dayfile.

DEVICE UNAVAILABLE, AT nnn.	This message is issued if the user's master device† was not transferred to the backup system.
DIRECT ACCESS DEVICE ERROR, AT nnn.	This message is issued if the user attempted to reference direct access files on a device (other than his master device) not present in the backup system.†

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

FILE MANAGEMENT CONTROL STATEMENTS 7

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

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

The statements in this section allow the user to position his files, copy data from one file to another, specify method and format of input/output, sort his files, and add corrections. He can assign his files to a specific device type; change the file type, identification code, and write interlock status; and release them from job attachment. The user can also receive information about records in a file or documentation in a file containing COMPASS source code. The statements do not use CRM.

If a file is not specifically assigned through the use of an ASSIGN, LABEL, or REQUEST control statement, the system assigns the file to available mass storage.

ASSIGN STATEMENT

The ASSIGN control statement directs the system to assign a file to the specified device or device type. The following descriptions refer to devices other than magnetic tape. For use of the ASSIGN statement with magnetic tape, refer to section 10.

The control statement format is:

ASSIGN(nn,lfn, { CK / CB })

nn — Device or device type to which the specified file is to be assigned; nn may be either the EST ordinal † of a peripheral device or the device type as defined as follows:

Type	Equipment
DE	Extended core storage
DI	844-21 Disk Storage Subsystem (half-track)
DJ	844-41 or 844-44 Disk Storage Subsystem (half-track)
DK	844-21 Disk Storage Subsystem (full-track)
DL	844-41 or 844-44 Disk Storage Subsystem (full-track)
DM	885 Disk Storage Subsystem (half-track)
DP	Distributive data path to ECS
DQ	885 Disk Storage Subsystem (full-track)
MS	Mass storage device
NE	Null equipment
TT	Time-sharing terminals††

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

CK or CB — Specifies that lfn is to be used as a checkpoint file (refer to section 11).

CK	Each dump is written at the previous EOI of lfn.
CB	Each dump is written at the BOI of lfn.

Example 1:

```
ASSIGN(MS,OUTPUT)
```

This statement assigns file OUTPUT to mass storage. With this assignment, a time-sharing user causes output normally printed at his terminal to be written on a mass storage file instead. Here, output means information generated by a program during execution. Dayfile messages are still printed at the terminal. Once this assignment is made, output is written on the mass storage file OUTPUT until the file is returned or reassigned.

---

†Contact installation personnel for a list of EST ordinals.
††This device type applies only to time-sharing origin jobs.

Example 2:

```
ASSIGN(TT,XYZ)
```

This statement assigns file XYZ to the user's time-sharing terminal. The assignment means that input that the system would have read from file XYZ is instead solicited by a prompt at the terminal and that output that the system would have written on file XYZ is instead displayed at the terminal.

Exammple 3:

```
ASSIGN(DI,ABC)
```

This statement assigns file ABC to an 844-21 disk drive, if one is available.

The ASSIGN statement can also be used to create or access existing seven- or nine-track unlabeled tapes. For a description of the statement as it applies to magnetic tape assignment, refer to Tape Management in section 10.

BKSP STATEMENT

The BKSP control statement directs the system to bypass a specified number of logical records in the reverse direction.

The control statement format is:

```
BKSP(lfn,n,m)
```

lfn	Name of the file to be backspaced.
n	Number of logical records (decimal) to backspace; if this parameter is omitted, the system assumes n=1.
m	File mode: C for coded, B for binary. If m is omitted, the system assumes the file is in binary mode.

The BKSP request can be issued at any point in a logical record. If, for example, FILE1 were positioned within the third record, a

```
BKSP(FILE1)
```

request would reposition FILE1 to the beginning of the third record. The system does not backspace past the beginning-of-information (BOI) or load point (tape file). However, EOF indicators are considered separate records and are included in the record count. An unrecognizable record count causes the message.

```
ERROR IN FILE ARGUMENTS.
```

to be issued to the user's dayfile.

The BKSP statement has no effect on a primary file since that file is rewound before every operation.

CLEAR STATEMENT

The CLEAR control statement releases all files currently assigned to the job. The user can also specify files that are not to be released.

The control statement formats are:

CLEAR.

or

CLEAR(*,lfn_1,lfn_2,...,lfn_n)

The first format releases all files. The second format releases all files except those named. If no files are named, all files assigned to the job are released.

If the CLEAR control statement is included in a CCL procedure, the CCL work files (ZZZZZC0, ZZZZZC1, and ZZZZZC2) are not returned. When the CLEAR control statement is not within a CCL procedure, the CCL work files are returned, unless they are specified in the second format of the CLEAR control statement.

Refer to RETURN statement in this section for the operations performed on each file type.

COMMON STATEMENT

The COMMON control statement creates or accesses a library type file (LIFT).

The control statement format is:

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

lfn Logical file name.

The user must be validated to access or create library files. The specified file must be a local mass storage file. If lfn is not local, a search is made for a library file by that name, and an error message is issued if the file is not found. If the operation completes successfully, the file is attached to the user's job as a library type file.

Before a local file can be made a library file, it must be locked. Refer to LOCK Statement in this section.

CONVERT STATEMENT

The CONVERT control statement converts records from one character set to another.

The control statement format is:

CONVERT(p_1,p_2,...,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 n 500. If omitted, 300 is the assumed maximum record size. (Each character is 6 bits.)

64 Convert from 63- to 64-character set; if omitted, no conversion takes place. The TS option must be specified if 64 is not.

TS=t Convert from old time-sharing 61-character set to new time-sharing 63-character set; t may be one of the following terminal types.

t	Terminal Type
TTY	ASCII code terminal with standard print.
COR	Correspondence code terminal with standard print.
CORAPL	Correspondence code terminal with APL print.
MEMAPL	Memorex 1240 (ASCII code) terminal with APL print.
BLKEDT	Block transmission (ASCII code) terminal with full display screen editing capability and standard print.
NAMIAF	Virtual network terminal. Same as TTY.

If t is omitted, it is assumed to be TTY. If TS is omitted, no time-sharing conversion takes place. The 64 option must be specified if TS is not.

R Rewind input and output files before, but not after, processing. If omitted, the files are not rewound before or after processing.

RC=n_2 Convert n_2 decimal records. If n_2 is omitted, convert until an EOF is encountered. If RC=n_2 is omitted, one record is assumed.

NM — Used in conjunction with TS parameter and specifies that conversion is to normal mode; if omitted, conversion is to ASCII mode. Note the effect of conversion on the following characters.

^	(circumflex)	If TS is specified, display code 70 (circumflex character) is converted to 76. If NM is omitted, conversion is to 7402 (ASCII mode).
:	(colon)	If TS and 64 are specified, display code 63 (colon character) is converted to 00. If NM is omitted, conversion is to 7404 (ASCII mode).

The following lists legal conversion using the appropriate CONVERT parameter.

Type of Record	Legal Conversion Parameters
63-character set, nontime-sharing record	64
Old time-sharing record	TS or 64 and TS
New NORMAL time-sharing record (equivalent to BATCH character set)	64
New ASCII time-sharing record	None

COPY STATEMENT

The COPY control statement copies data from one file to another if the files are within the range of permissible formats listed in table 1-7-1.

TABLE 1-7-1. RANGE OF PERMISSIBLE FORMATS FOR THE COPY STATEMENT

Input (I=lfn$_1$)		Output (O=lfn$_2$)					
		Mass Storage or Terminal	Tape Formats				
			I	SI	S	L	F
Mass Storage or Terminal		Yes	Yes	Yes	Yes	Yes	No
Tape Formats	I	Yes	Yes	Yes	Yes	Yes	No
	SI	Yes	Yes	Yes	Yes	Yes	No
	S	Yes	Yes	Yes	Yes	Yes	No
	L	Yes	Yes	Yes	No	Yes	No
	F	Yes	Yes	Yes	No	No	Yes

The parameters can appear in order dependent format, order independent format, or a combination of both. The completely order dependent format is:

COPY(lfn_1,lfn_2,x,c,tc,copycnt,bsize,charcnt,erlimit,$p_1p_2...p_n$,lfn_3)

The completely order independent format is:

COPY(I=lfn_1,O=lfn_2,V=x,M=c,TC=tc,N=copycnt,BS=bsize,CC=charcnt,EL=erlimit,
PO=$p_1p_2...p_n$,L=lfn_3)

If order dependent and order independent parameters are mixed in one COPY statement, the order dependent parameters must appear in their proper position. All parameters are optional. However, the specification of certain parameters precludes the application of others. A nonapplicable parameter may be ignored or it may be illegal. This is stated in the individual descriptions of the parameters.

The parameters are defined as follows:

Parameter	Description	Default
I=lfn_1	Name of the file to copy from.	INPUT
O=lfn_2	Name of the file to copy to.	OUTPUT
V=x	If the x parameter (one to seven alphanumeric characters) is present, both files are rewound, copied, rewound, verified, and rewound. The x parameter must not be zero.	No verify
M=c	M=C1 Coded mode is set on input only. M=C2 Coded mode is set on output only. M=any other value (one to seven alphanumeric characters) Coded mode is set on both input and output. This parameter applies only to S and L format tapes. If coded mode is set on an SI tape, the system aborts the job. For other formats, the system ignores the mode setting.	Binary
TC=tc	Specifies the copy termination condition used in conjunction with N=copycnt. The termination condition can be specified as follows: tc — Meaning F or EOF — The N keyword specifies the number of files to copy. I or EOI — Copy to the end of information. The N keyword is ignored.	Copy to double EOF (TC=D or TC=EOD)

Parameter	Description	Default
	tc — Meaning	
	D or EOD — The N keyword is the number of double EOFs to copy to. If N>1 is specified together with this TC value, and verify is also selected, the files are verified only to the first empty file (COPY calls VERIFY with N=0 parameter).	
N=copycnt	Copy count used with the copy termination condition specified by the parameter TC.	1
BS=bsize	Maximum block size (in central memory words) which specifies S or L tape PRU size. This applies only when copying to or from S and L tapes. It cannot be specified with the CC parameter.	If CC is not specified, 1000_8 for S tape copy and 2000_8 for L tape copy.
CC=charcnt	Maximum number of characters in an S or L tape block. This parameter can be specified only when copying to or from S and L tapes. The PRU size and unused bit count are calculated from the character count. However, the unused bit count is used only when writing a full block to an S or L output tape during a copy from mass storage, I, or SI format tape. The charcnt value should be a multiple of 10. If it is not, the characters that exceed the charcnt value in the last word of the record are discarded when writing an S or L format tape. This parameter cannot be specified with the BS parameter.	Not used (the PRU size is specified by the BS parameter)
EL=erlimit	Error limit which specifies the number of non-fatal errors allowed before abort. This includes both parity errors and block-too-large errors which are returned by the tape subsystem after completing recovery procedures. If EL=U is specified, unlimited error processing is allowed. Error recovery is supported on mass storage and on all tape formats but is not supported on a terminal or on unit record equipment. In the latter cases, any error aborts the job.	0 (zero)
PO=$p_1p_2 \cdots p_n$	One or more of the following processing options:	
	E — Input blocks with parity errors or block-too-large errors are processed (copied).	Error blocks are skipped.
	D — Any noise blocks generated by a copy from mass storage, I format tape, or SI format tape to an S or L format tape are deleted. This parameter cannot be specified on any other type of copy.	For S or L binary tapes, noise blocks are padded to noise size with binary zeros; for coded mode, they are padded with blanks.

Parameter		Description	Default
	R	Allows record splitting during a copy from mass storage, I format, or SI format to S or L format tape. This parameter cannot be specified on any other type of copy.	Record splitting is not allowed.
	M	Copy files according to the copy termination condition specified by the keyword TC, eliminating each EOF on output. This option is primarily for use with labeled S and L output tapes since it eliminates the conflict of the double meaning of a tape mark on these formats (the tape mark on these formats serves as both an EOF and label group delimiter).	Copy files according to specification of the copy termination (TC), writing an EOF after each file on output.
$L=lfn_3$		Name of an alternate output file to receive parity error messages when extended error processing is in effect (nonzero EL specified), in which case, the file name lfn_3 must not be the same as lfn_1 or lfn_2.	OUTPUT

Example:

The following COPY statement combines order dependent and order independent parameters.

```
COPY(FILE1,FILE2,VERIFY,CODED,EOF,6,L=MYOUT,PO=E,EL=10)
```

FILE1 is the input file, and FILE2 the output file. Six coded files are copied and verified. Up to 10 nonfatal errors are allowed, and the bad data is copied with informative error messages written to the file MYOUT.

The COPY statement begins a copy operation at the current position of both files unless the verify option is specified. If verify is specified, both files are rewound before the copy begins and rewound, verified, and rewound again after the copy is completed. (This verify may not be meaningful if the logical structure of the two files is incompatible.)

Copy Termination

Copying continues until the copy termination condition is met or EOI is encountered. The copy termination condition can be a file count, a double EOF count, or EOI. If the copy is terminated by a double EOF (for TC=EOD option), the second EOF is detected on lfn_1, but is not transferred to lfn_2. If lfn_1=lfn_2 the named file is read until the termination condition is satisfied or EOI is encountered.

If a copy specifies a file count, TC=EOF, and EOI is encountered on the input file before the file count is satisfied, an additional EOF is written on the output file only if data or records have been transferred since the previous EOF was written (or since the beginning of the copy if no EOFs have been encountered).

Block Sizes

Both L and F tapes may require additional field length to accommodate their maximum block size. The maximum block size for an L tape copy is specified either by the BS=keyword (or its default), or it is calculated from the CC=keyword. The maximum block size for an F tape is determined by the maximum frame or character count specified when the file was assigned. The more accurate the selection of these values which determine block size, the less are the requirements for field length, CPU time, and I/O time.

Processing Options

The PO=D option specifies noise block processing, and the PO=R option specifies record splitting for copies from mass storage, I format, or SI format to S or L format tapes. Due to the incompatibilities between the logical structure of the input and output files, records may be encountered on the input file that are too small or too large to be copied directly to the S or L output tape. If the output file block size is less than noise block size, it is deleted if PO=D is specified. If PO=D is not specified, the block size is rounded to the word multiple of noise size with binary zero fill for a binary S or L tape or with blank fill for a coded S or L tape. Empty records on the input file are skipped since they cannot exist on an S or L tape. If PO=R is specified and an input file record length exceeds the S or L tape maximum block size (the PRU size as specified by BS= or its default, or by CC=), it is split into multiple blocks. If PO=R is not specified and an input record length exceeds the S or L tape maximum block size, the job aborts with the message

RECORD TOO LARGE ON lfn.

The PO=M option makes it possible to copy a multifile file to a labeled S or L format tape without writing the EOF tape marks. This avoids the conflict of a tape mark serving the double purpose of defining an EOF and delimiting a label group on S and L format tapes. This is in keeping with the tendency in the computer industry to define a tape mark only as a label delimiter.

The EL and PO=E options provide extended error processing. If EL is set to a value greater than zero, a parity error or a block-too-large error on the input file generates the following message on the alternate output file.

PARITY/BLOCK TOO LARGE ERROR IN BLOCK n.

n is the decimal block count of the block in error.

COPYBF STATEMENT

The COPYBF control statement copies a specified number of files from one multifile file to another.

NOTE

The COPYBF statement is not recommended for use with S, L, or F format tapes because it does not have the data specification parameters needed to accommodate the variety of data formats possible with those tape formats. For S, L, or F format tape copy operations, the user should issue a COPY statement with the appropriate parameter specifications.

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	Fourth parameter (one to seven alphanumeric characters) indicating that the copy to or from an S or L format tape should be performed in coded rather than binary mode. If coded mode is set on an SI tape, the system aborts the job. The system ignores this parameter for mass storage files and I and F format tape files.

The copy begins at the current position of lfn_1. If lfn_1=lfn_2, the file is read until the file count is satisfied or EOI is encountered.

If EOI is encountered on lfn_1 before the file count is satisfied, an additional EOF is generated on lfn_2 only if data or records have been transferred since the previous EOF was written (or since the beginning of copy if no EOFs have been encountered).

COPYBR STATEMENT

The COPYBR control statement copies a specified number of records from one file to another.

NOTE

The COPYBR statement is not recommended for use with S, L, or F format tapes because it does not have the data specification parameters needed to accommodate the variety of data formats possible with those tape formats. For an S, L, or F format tape copy operation, the user should issue a COPY statement with the appropriate parameter specifications.

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	Fourth parameter (one to seven alphanumeric characters) indicating that the copy to or from an S or L format tape should be performed in coded rather than binary mode. If coded mode is set on an SI tape, the system aborts the job. The system ignores the mode setting for other formats.

The copy begins at the current position of lfn_1. EOF indicators are considered separate records and are included in the record count. If lfn_1=lfn_2, the file is read until the record count is satisfied or EOI is encountered.

If EOI is encountered on lfn_1 before the record count is satisfied, an additional EOR is written on lfn_2 only if data has been transferred since the previous EOR or EOF was written (or since the beginning of the copy if no EORs or EOFs have been encountered).

COPYCF STATEMENT

The COPYCF control statement copies a specified number of coded files from one file to another. A coded file is defined as a file containing lines of 150 characters or less, each terminated by a zero byte (12 zero bits in the lowest byte of a word).

NOTE

The COPYCF statement is not recommended for use with S, L, or F format tapes because it does not have the data specification parameters needed to accommodate the variety of data formats possible with those tape formats. For an S, L, or F format tape copy operation, the user should issue a COPY statement with the appropriate parameter specifications.

The COPYCF statement cannot copy SI format tapes. If coded mode is set for an SI tape, the system terminates the job. The TCOPY utility converts SI coded tape files.

The control statement format is:

COPYCF(lfn_1,lfn_2,n,fchar,lchar,na)

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 (1 to 150); if this parameter is omitted, the copy begins at character position 1.†
lchar	Last 6-bit character position of each line to copy (1 to 150); lchar must be greater than or equal to fchar. If this parameter is omitted,† the copy ends at character position 136.
na	Sixth parameter (one to seven alphanumeric characters) specifying that the job step should not abort when a line terminator does not appear before an EOR.

The copy begins at the current position of lfn_1. If lfn_1=lfn_2, the file is read until the file count is satisfied or EOI is encountered. If EOI is encountered before the file count is satisfied, an EOF is written on lfn_2, and the operation terminates. If a line is encountered that has more than lchars, the excess characters are truncated.

COPYCF writes lines with an even number of characters. If an input line has an odd character count and the last character is a blank not immediately preceded by a colon, the last character is removed. If an input line has an odd character count and the last character is not a blank or is a blank immediately preceded by a colon, an additional trailing blank is appended.

If COPYCF attempts to copy a line longer than 150 6-bit characters, the line is truncated, and an informative message is issued to the dayfile after the copy completes.

If the last line of a record does not have an end-of-line terminator, COPYCF issues a dayfile message. If the na parameter is not specified, the job step then aborts.

† Since many characters in 6/12 display code require 12 bits rather than 6, this parameter may produce unforeseen results when copying a file containing 6/12 display code data.

COPYCR STATEMENT

The COPYCR control statement copies a specified number of coded records from one file to another. A coded record contains lines of 150 characters or less, each terminated by a zero byte (12 zero bits in the lowest byte of a word).

NOTE

The COPYCR statement is not recommended for use with S, L, or F format tapes because it does not have the data specification parameters needed to accommodate the variety of data formats possible with those tape formats. For an S, L, or F format tape copy operation, the user should issue a COPY statement with the appropriate parameter specifications.

The COPYCR statement cannot copy SI format tapes. If coded mode is set for an SI tape, the system terminates the job. The TCOPY utility converts SI coded tape files.

The control statement format is:

COPYCR(lfn_1,lfn_2,n,fchar,lchar,na)

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 (1 to 150); if this parameter is omitted, the copy begins at character position 1.†
lchar	Last 6-bit character position of each line to copy (1 to 150), lchar must be greater than or equal to fchar. If this parameter is omitted, the copy ends at character position 136.†
na	Sixth parameter (one to seven alphanumeric characters) specifying that the job step should not abort if a line terminator does not appear before an EOR.

The copy begins at the current position of lfn_1. If lfn_1=lfn_2, the file is read until the record count is satisfied or EOI is encountered. EOF indicators are considered separate records and are included in the record count. If the EOI is encountered before the record count is satisfied, an EOF is written on lfn_2, and the operation terminates. COPYCR is processed in exactly the same manner as the COPYCF control statement except that n specifies the number of records rather than the number of files.

† Since many characters in the 6/12 display code require 12 bits rather than 6, this parameter may produce unforeseen results when copying a file containing 6/12 display code data.

If COPYCR attempts to copy a line longer than 150 6-bit characters, the line is truncated, and an informative message is issued to the dayfile after the copy completes.

If the last line of a record does not have an end-of-line terminator, COPYCR issues a dayfile message. If the na parameter is not specified, the job step then aborts.

COPYEI STATEMENT

The COPYEI control statement copies one file to another. The copy begins at the current position of the file and continues until the EOI is encountered. The EOI is not defined for certain tape formats (refer to table 1-2-1).

NOTE

The COPYEI statement is not recommended for use with S, L, or F format tapes because it does not have the data specification parameters needed to accommodate the variety of data formats possible with those tape formats. For an S, L, or F format tape copy operation, the user should issue a COPY statement with the appropriate parameter specifications.

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 (one to seven alphanumeric characters) is present, both files are rewound before the copy, and rewound, verified, and rewound again after the copy is complete.
c	Fourth parameter (one to seven alphanumeric characters) indicating that the copy to or from an S or L format tape should be performed in coded rather than binary mode. If coded mode is set on an SI tape, the system aborts the job. For other formats, the system ignores the mode setting.

If lfn$_1$=lfn$_2$, the file is read until EOI is encountered.

COPYSBF STATEMENT

The COPYSBF control statement enables the user to copy a file where the first character of each line is not a printer control character and is to be printed.

NOTE

The COPYSBF statement is not recommended for use with S, L, or F format tapes because it does not have the data specification parameters needed to accommodate the variety of data formats possible with those tape formats. For an S, L, or F format tape copy operation, the user should issue a COPY statement with the appropriate parameter specifications.

The control statement format is:

COPYSBF(lfn_1,lfn_2,n,na)

lfn_1	Name of the file to copy from; if this parameter is ommitted, 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.
na	Fourth parameter (one to seven alphanumeric characters) specifying that the job step should not abort if a line terminator does not appear before an EOR.

The COPYSBF routine copies n files beginning at the current position of lfn_1 to file lfn_2, shifting each line image one character to the right and adding a leading space. Each line image may contain up to 150 (6-bit) characters. Any characters beyond 150 are lost. A page eject character is inserted at the beginning of each logical record (refer to appendix I for a list of carriage control characters). If lfn_1=lfn_2, n files are skipped but no data transfer occurs. If the EOI is encountered before the file count is satisfied, an EOF is written to lfn_2, and the operation terminates.

If COPYSBF attempts to copy a line longer than 150 6-bit characters, the line is truncated, and an informative message is issued to the dayfile after the copy completes.

If the last line of a record does not have an end-of-line terminator, COPYSBF issues a dayfile message. If the na parameter is not specified, the job step then aborts.

COPYX STATEMENT

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

NOTE

The COPYX statement is not recommended for use with S, L, or F format tapes because it does not have the data specification parameters needed to accommodate the variety of data formats possible with those tape formats. For an S, L, or F format tape copy operation, the user should issue a COPY statement with the appropriate parameter specifications.

The control statement format is:

COPYX(lfn_1,lfn_2,x,b,c)

lfn_1 Name of the file to copy from; if this parameter is omitted, file INPUT is assumed.

lfn_2 Name of the file to copy to; if this parameter is omitted, file OUTPUT is assumed.

x Copy specifications; if omitted, one record is copied. The value for x may be one of the following:

x	Meaning
n	Number of records (decimal) to copy.
00	Copy all records up to and including first zero-length record.
name	Copy all records up to and including record of specified name (record name is first seven characters of record or the name in the prefix table, if present).
type/name	Copy all records up to and including record of specified type and name (refer to Library Record Types in section 14 for list of valid record types).

b Backspace control; if omitted, 0 is assumed.

b	Meaning
0	No backspace.
1	Backspace file lfn_1 one record after copy completes.
2	Backspace file lfn_2 one record after copy completes.
3	Backspace files lfn_1 and lfn_2 one record after copy completes.

c — Fifth parameter (one to seven alphanumeric characters) indicating that the copy to or from an S or L format tape should be performed in coded rather than binary mode. If coded mode is set on an SI tape, the system aborts the job. For mass storage files and I and F format tape files, the mode setting is ignored.

The COPYX routine copies logical records from lfn_1 to file lfn_2 at the current position of lfn_1 until the condition specified by x is met. It then backspaces the files according to the value specified by the b parameter. If an EOF or EOI is encountered on lfn_1 before the condition specified by x is met, the operation terminates and the backspace parameter b is ignored. If lfn_1=lfn_2, the file is read until the termination condition is satisfied or an EOF or EOI is encountered.

If EOI is encountered on lfn_1 before the termination condition is satisfied, an additional EOR is written on lfn_2 only if data has been transferred since the previous EOR was written (or since the beginning of the copy if no EORs have been encountered).

DISPOSE STATEMENT†

The DISPOSE control statement releases the specified files to the named output queues.

The control statement format is:

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

lfn_i — Name of the file to be disposed. lfn cannot be a direct access file or the primary file.

q_i — Queue type:

PR	Print
PH	Punch coded O26
P9	Punch coded O29
PB	Punch binary
P8	Punch 80-column binary

ot — Origin type to which files are to be disposed:

BC	Local batch
EI	Remote batch

usernum — Number of the remote batch user to which the files are to be disposed (ignored if ot is BC). 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.

† The user should employ the ROUTE control statement for this operation, because the DISPOSE statement will not be supported in future versions of NOS.

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 devices. If ot is BC, the usernum parameter is ignored, and the files are disposed to the central site devices.

DOCMENT STATEMENT

The DOCMENT control statement enables the user to extract either the external or internal documentation from a file.

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 I in volume 2 for a detailed explanation of the documentation standards followed. This appendix also contains an example of external and internal documentation for a sample program.

EVICT STATEMENT

The EVICT control statement releases file space for the specified files but does not release file assignment to the job.

The control statement format is:

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

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

The operation that EVICT performs depends on the file characteristics.

File	EVICT Action
Permanent and primary files	Releases all file space except the first track and writes an EOI on the first sector of the first track, but keeps file assigned to the job.
Deferred routed queue file †	Releases all file space and clears all file routing information.
File with write interlock set	Unloads file.
Tape files	Releases tape from the job.
All other files	Releases file space, but keeps file assigned to job.

An EVICT of a tape file performs the same function as an UNLOAD and so cannot be used to decrease the number of resource units scheduled via the RESOURC statement.

FCOPY STATEMENT

The FCOPY control statement converts a file from one code set to another. Currently, the only supported conversion is from 6/12 display code (used in time-sharing ASCII mode) to 12-bit ASCII code. Refer to appendix A for code set definitions.

The control statement format is:

FCOPY(P=lfn_1,N=lfn_2,PC=cs_1,NC=cs_2,R)

P=lfn_1	File to be converted (default is OLD). The user should assign lfn_1 to the job before performing the FCOPY operation.
N=lfn_2	File on which the converted data from lfn_1 is written (default is NEW). If lfn_2 is not assigned to the job, FCOPY creates it.
PC=cs_1	Code set of lfn_1. The default and only currently supported value for cs_1 is ASCII, which refers to 6/12 display code.
NC=cs_2	Code set of lfn_2. The default and only currently supported value for cs_2 is ASCII8, which refers to 12-bit ASCII code.
R	If R is specified, lfn_1 and lfn_2 are rewound before and after the conversion. If R is omitted, lfn_1 and lfn_2 are not rewound before or after the conversion.

† Refer to the ROUTE statement in this section.

FCOPY reads lfn_1 to its EOI, preserving its EOR and EOF marks on the converted file. The maximum line length that can be processed is 160 12-bit codes or 320 6-bit codes. Lines that exceed the maximum length are truncated.

NOTE

If lfn_1 is written in 6/12 display code based on the 63-character set, it must be converted to the 64-character set by the CONVERT control statement before its conversion by the FCOPY statement.

Files converted to 12-bit ASCII code can be listed on a local batch printer (refer to the ROUTE control statement) but cannot be listed at a time-sharing or remote batch terminal.

Example:

A time-sharing user wants to print a file (FILE1) created in ASCII mode. To do so, he enters a COPYSBF statement to prefix the file lines with appropriate carriage control characters. He then enters an FCOPY statement to convert the file containing 6/12 display code (FILE2) to a file containing 12-bit ASCII code (FILE3). Finally, he routes the converted file (FILE3) to a line printer that prints the ASCII graphic 95-character set.

```
/ascii
/copy,file1.
AaBbCcDdEeFfGg
HhIiJjKkLlMmNn
 EOI ENCOUNTERED.
/rewind,file1.
 $REWIND,FILE1.
/copysbf,file1,file2.
 END OF INFORMATION ENCOUNTERED.
/rewind,file2.
 $REWIND,FILE2.
/copy,file2.
1AaBbCcDdEeFfGg
 HhIiJjKkLlMmNn
 EOI ENCOUNTERED.
/fcopy,p=file2,n=file3,r.
 FCOPY COMPLETE.
/route,file3,dc=lp,ec=a9.
 ROUTE COMPLETE.
```

The following is the local batch output from the ROUTE statement.

```
AaBbCcDdEeFfGg
HhIiJjKkLlMmNn
```

LIST80 STATEMENT

The LIST80 routine reads a file containing list output produced by the COMPASS assembler and compresses it to 80 columns, which fits on 8-1/2- by 11-inch printer paper.

The control statement format is:

LIST80(lfn_1,lfn_2,NR)

lfn_1	File to copy from; if this parameter is omitted, file LIST is assumed.
lfn_2	File to copy to; if this parameter is omitted, file OUTPUT is assumed.
NR	Parameter indicating that lfn_1 should not be rewound.

The LIST80 statement output listing omits the following information that appears on the COMPASS assembler output listing.

- COMPASS version number on the page heading.
- COMPASS assembler binary values.
- INVENTED SYMBOLS comment.
- Symbolic Reference Table block column that contains either the system text file name, the overlay name, or the name of the block containing the symbol.

Comments are truncated to column 65.

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.

x	Description
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 six fields).

x(1 to 6) — Number of field being moved.

y — Number of characters being moved.

NOTE

The following restrictions apply to the H, N, I, and O parameters.

(Nx+Ix).GT.150	Yields an error (I≤x≤6).
(Nx+Ox).GT.H	Yields an error (I≤x≤6).
H.GT.150	Yields an error.

Ix=y — Specifies the field the data originates from.

x(1 to 6) — Number of 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 wants to change any of the input parameters before processing begins. If omitted, query is issued. This parameter is effective only from time-sharing origin jobs.

The following shows the default values assumed for the N, O, and I parameters for the various source types.

Type	N1	I1	O1	N2	I2	O2	N3	I3	O3
B	72	1	1	0	0	0	0	0	0
C	7	9	1	50	41	8	15	112	58
M	2	6	1	48	10	3	22	82	51

The remaining parameters of these types are defaulted to 0.

LO72 reformats files (output files in general). The user can rearrange each line (all lines must be formatted the same) in the format he chooses. All default values compress output to 72 columns, which is appropriate for terminal output or 8-1/2- by 11-inch printer paper. If a 1 is encountered in column 1 (the page eject printer control character), the next two lines of source data are processed as a two-line header. This header is compressed to 72 columns for all source types. If no page eject control characters are encountered, no headers are processed.

The following values apply to the first line of header and cannot be changed.

N1=42, I1=8, O1=0 (if LP not specified; otherwise, O1=1).

N2=20, I2=90, O2=42.

N3=5, I3=115, O3=62.

N4=5, I4=121, O4=67.

The subheader lines for COMPASS and Modify listings are processed uniquely.

For B listings, the following values apply to the reformatting.

N1=43, I1=8, O1=0 (if LP not specified; otherwise, O1=1).

N2=29, I2=70, O2=43.

All parameters are passed to LO72 by the control statement. If an input file is specified, LO72 reads it for additional input parameters. If the job originates from a time-sharing terminal, and the IT parameter is not specified, the user is asked if he wishes to change any of the input parameters. If he enters YES, the system prints the current parameter values and allows him to change them individually. Pressing the carriage return key for any parameter leaves the parameter at its former value. In the following examples, the same input parameters are entered in three possible ways.

Examples:

Control Statement:

```
LO72(I=0,S=SOURCE,T=B,L=OUT,N4=1,I4=2,O4=75,H=90)
```

Time-Sharing Terminal:

```
/lo72
DO YOU WANT TO CHANGE ANY CONTROL APGUMENT VALUES-
ENTER: YES OR NO
? yes (CR)
ARGUMENT              VALUE
INPUT FILE NAME:              ? (CR)
SOURCE FILE NAME:    SCR      ? source (CR)
OUTPUT FILE NAME:    OUTPUT   ? out (CR)
SOURCE FILE TYPE:    BATCH ? b (CR)
OUTPUT LINE LENGTH: 72  CHARS.? 90 (CR)
   NO. OF  MOVED FROM  MOVED TO
   CHARS.    COLUMN     COLUMN
(X) (NX)      (IX)       (OX)
 1.  72        1          1
 2.  0         0          0
 3.  0         0          0
 4.  0         0          0
 5.  0         0          0
 6.  0         0          0
ENTER CHANGES IN THE FOLLOWING FORMAT:
NX=AA*CR*
IX=BB*CR*
OX=CC*CR*
ETC.
TO CONTINUE, ENTER *CR* ONLY. ? n4=1 (CR)
? i4=2 (CR)
? o4=75 (CR)
? (CR)
 LO72 COMPLETE.
```

User entries are in lowercase. The symbol ⓒⓡ indicates carriage return.

Input File:

```
S=SOURCE, L=OUT, T=B.
N4=1, I4=2, O4=75.
H=90.
-EOR-
```

Each line in the input file must end with a terminator.

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, no files currently assigned to the job are released.

The NEW statement creates an empty file and makes it the user's new primary file. All files assigned to the job are released unless the ND parameter is specified. When the ND parameter is specified, any currently existing primary file becomes a nonprimary temporary file.

The automatic rewinding of primary files is incompatible with some file manipulation statements. Refer to the note in PRIMARY Statement later in this section.

OUT STATEMENT

The OUT control statement releases output files to the output (punch or print) queue.

The control statement format is:

OUT.

or

OUT(*,lfn_1,lfn_2,...,lfn_n)

The first format releases all eligible files to the appropriate output queues. The second format releases all eligible files except those named (lfn_i). If no files are named, all eligible files are released.

Files eligible for release are:

- Print (PRFT) type.
- Punch (PHFT) type.
- Local (LOFT) type, if named OUTPUT, PUNCH, PUNCHB, or P8.

The file type determines the output queue to which the file is sent. For example, a print type file named PUNCH is released to the print queue and a punch type file named PUNCHB is released to the punch queue.

The number of files released is recorded in the job's dayfile with the message

nnn FILE(S) PROCESSED.

nnn is the decimal number of files to be released.

If no files with the above names or belonging to these types are found, the following message is issued to the dayfile

NO FILE(S) RELEASED.

This control statement is used if the user wishes to initiate printing or punching of the files before job termination. The PUNCH file is punched in either O26 or O29 mode, depending on the origin of the job. If the job is a local batch job, the coded deck is punched in the initial keypunch mode of the job's control statement record. For all other job origin types, the coded file is punched in the system default keypunch mode.

PACK STATEMENT

The PACK control statement removes all EOR and EOF marks from a specified file and copies it as one record to another file.

The control statement format is:

PACK(lfn_1,lfn_2,x)

lfn_1	Name of file to be packed. From batch origin, lfn_1 must be specified; from time-sharing origin, the primary file is used if lfn_1 is omitted. The file must not be assigned to a time-sharing terminal (file type TT).
lfn_2	Name of file to receive packed data. If lfn_2 is omitted, the packed file is written on lfn_1.
x	Third parameter (one to seven alphanumeric characters) indicating that lfn_1 should not be 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. The results of PACK may produce unpredictable results when used with certain CYBER Record Manager (CRM) files (refer to CRM Basic Access Methods or CRM Advanced Access Methods Reference Manual).

PRIMARY STATEMENT

The PRIMARY control statement makes a local file the primary file, or it creates an empty primary file.

The control statement format is:

PRIMARY(lfn)

lfn	Name of local file.

If lfn already exists, it must be a local mass storage file in order to be made the primary file. If lfn does not exist, the PRIMARY statement creates it on mass storage. Any currently existing primary file (other than the lfn specified) becomes a nonprimary temporary file. If the specified file is already primary, the operation is ignored.

NOTE

The primary file is rewound before every operation performed on that file. Therefore, the file manipulation statements BKSP, SKIPEI, SKIPF, SKIPFB, and SKIPR cannot be used to position within the file. The user should also remember that the primary file is rewound after the completion of any of the COPY statements. An attempt to add to the file using one of the COPY statements may result in writing over existing data at the BOI.

RENAME STATEMENT

The RENAME control statement allows the user to change the name of a local file.

The control statement format is:

RENAME($nlfn_1$=$olfn_1$,$nlfn_2$=$olfn_2$,...,$nlfn_n$=$olfn_n$)

$nlfn_i$ New name of the local file.

$olfn_i$ Existing name of the local file.

The RENAME control statement changes the name of the file $olfn_i$ to $nlfn_i$ in the FNT/FST. This does not change the names of files in the permanent file system. Normaly, 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 released. Under certain conditions, the system also changes the file type of $olfn_i$ to that of the file which was released.

- If $olfn_i$ is a local mass storage file and the released file was a print, punch, or primary type file, $olfn_i$ is renamed and its file type is changed to that of the released file.
- If $olfn_i$ is a local mass storage file and the released file was not a print, punch, or a primary type file, $olfn_i$ is renamed but its file type is not changed.
- If $olfn_i$ is not a local file and nlfn and olfn are not the same file types or if $olfn_i$ does not reside on mass storage, an

  ILLEGAL FILE TYPE.

  error message is issued.

For example, the user has only two files assigned to the 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 no longer exists; file A is renamed file B and changed to print type file.

REQUEST STATEMENT

The REQUEST statement assigns a file to receive checkpoint dumps, or sends a message to the system operator requesting that the named file be assigned to the device described in the comment field.

The control statement format is:

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

lfn	Name of the file to be assigned to the specified equipment.
CK	Specifies that lfn is to be used as a checkpoint file. Each time a checkpoint dump is taken, the new information is written at the previous EOI of lfn.
CB	Specifies that lfn is to be used as a checkpoint file. Each time a checkpoint dump is taken, the new information is written at the BOI of lfn.
comment	The comment is displayed at the system console. In the comment field, the user directs the operator to make the requested device assignment.

If the REQUEST statement is used to request assignment of file lfn to the equipment specified in the comment field, lfn must not be a local file.

If the REQUEST statement is used to assign lfn for checkpoint dumps, lfn must be a local file and 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 checkpoint file lfn:

  REQUEST(lfn,CK)

- Save the last checkpoint dump by writing each dump at the beginning of checkpoint file lfn:

  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 a checkpoint file has been detected when the first CKP statment 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.

If lfn is a local file 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 may use the REQUEST statement to assign a file to a mass storage device. However, the user must be validated to assign a file to a magnetic tape or auxiliary device.†† If the user does not have this validation and attempts to request a tape unit or auxiliary device, the system aborts the job.

The REQUEST statement can also be used to create or access existing seven- or nine-track unlabeled tapes. If a magnetic tape assignment is needed to satisfy a REQUEST, the MT or NT parameter should be specified. For a description of magnetic tape assignment with the REQUEST statement, refer to Tape Management in section 10.

RESEQ STATEMENT

The RESEQ control statement is used to resequence source files which have leading sequence numbers or to add sequence numbers to an unsequenced file. The RESEQ command for time-sharing users has a different parameter order (refer to the Network Products IAF Reference Manual or the NOS Time-Sharing User's Reference Manual).

The control statement format is:

RESEQ(lfn,t,xxx,yy)

lfn — Name of the sorted file to be resequenced. RESEQ does not sort lfn (refer to the SORT statement).

t — Type of file:

- B — BASIC source code.
- T — Text source information; a five-digit sequence number plus a blank is added at the beginning of each line; the file text, however, is not inspected.

† Any mass storage file used as a checkpoint file must have write permission.
†† Refer to LIMITS Statement in section 6.

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 inserted at the beginning of each line. If the line number encountered or required exceeds 99999, RESEQ issues an error message.

When resequencing a BASIC source program, the user must specify B for the file type parameter, t, so that RESEQ changes the line number references within the source statements. RESEQ supplies five-digit line numbers and line number references; excess surrounding blanks are used in the expansion of line number references.

Example:

File X contains the following BASIC source statements.

```
95 ON SGN(A)+2 GOTO 100,110,120       'COMMENT
100    PRINT "A IS NEGATIVE"
105       GOTO    130                 'COMMENT
110    PRINT "A IS ZERO"
115       GOTO    130                 'COMMENT
120    PRINT "A IS POSITIVE"
130 LET B=A+130
135 END
```

The following statement changes the contents of file X.

```
RESEQ(X,B,90,10)
```

The user then rewinds and lists file X.

```
00090 ON SGN(A)+2 GOTO 00100,00120,00140 'COMMENT
00100    PRINT "A IS NEGATIVE"
00110       GOTO  00150                  'COMMENT
00120    PRINT "A IS ZERO"
00130       GOTO  00150                  'COMMENT
00140    PRINT "A IS POSITIVE"
00150 LET B=A+130
00160 END
```

The RESEQ statement changes the line numbers and the line number references. Line numbers now begin at 90 and increment by 10. The comment on the first line is moved to the right to allow for the expanded line number references.

RETURN STATEMENT

The RETURN control statement releases files assigned to a job and may release file space depending on the file type.

The control statement formats are:

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

or

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

The first format returns the named files (lfn_1,lfn_2,...,lfn_n). The second format returns all files assigned to the job except the named files. In a CCL procedure the second format does not return the CCL work files (ZZZZZC0, ZZZZZC1, and ZZZZZC2). A RETURN control statement not in a CCL procedure will return these files. If no files are named on the second format the asterisk specification returns all files assigned to the job. An error message is returned if neither an asterisk nor a file name is specified.

RETURN performs the following operations according to the file type.

Type	Operation
Input	File name is changed to INPUT*. File space is not released (refer to Input File Control in section 3 for further information).
Print	File space is released, and the file is no longer assigned to the job. (The file is not printed.)
Punch	File space is released, and the file is no longer assigned to the job. (The file is not punched.)
Local	File space is released, and the file is no longer assigned to the job.
Primary	Same as Local.
System	File space remains, but the file is no longer assigned to the job.
Library	File space remains, but the file is no longer assigned to the job.
Direct access	File space remains, but the file is no longer attached to the job.
Tape	Tape is no longer assigned to the job.

In addition, the RETURN of a magnetic tape file or the RETURN of the user's last direct access file on an auxiliary removable disk pack decrements the resource demand count as scheduled by the RESOURC control statement if, and only if, the total concurrent resource demand (tapes and removable packs) is presently assigned.

To release a file without decrementing the resource demand count, the user can issue an UNLOAD statement. To release file space without releasing the file from the job, the user can issue an EVICT statement.

REWIND STATEMENT

The REWIND control statement rewinds files. A mass storage file is positioned at its BOI. An unlabeled tape file is positioned at its load point. A labeled tape file is positioned after the first HDR1 label for the file. If the labeled tape file begins on a previous volume, the system notifies the operator to mount that volume.

The control statement formats are:

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

or

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

The first format rewinds the named files (lfn_1,lfn_2,...,lfn_n). The second format rewinds all files assigned to the job except the named files. If no files are named on the second format, the asterisk specification rewinds all files assigned to the job.

If the previous operation on the magnetic tape file was a write, a REWIND statement causes the following operations to be performed.

- If the tape is ANSI labeled, the system writes a tape mark, an EOF1 label, and three tape marks and then rewinds the tape.
- If the tape is unlabeled and the data format specified on the ASSIGN, LABEL, or REQUEST statement is S, L, or F, the system writes four tape marks and then rewinds the tape.
- If the tape is unlabeled and the data format is I or SI, the system writes a tape mark, an EOF1 label, and three tape marks and then rewinds the tape.

Refer to Magnetic Tape Files in section 2 and to Tape Management in section 10 for further information about tape files and to appendix G for a description of EOF1 and EOV1 labels.

ROUTE STATEMENT

The ROUTE control statement prepares a designated file for release to an input or output queue. The file routing requested may take effect when the statement is processed, or it may be deferred. If deferred, the routing characteristics specified define the handling of the file in later job steps or at job termination. This statement also allows the user to rescind a prior deferred ROUTE statement, changing the file type to local. For a description of job names assigned to routed files, refer to Job Names in section 3.

The control statement format is:

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

Descriptions of the statement parameters follow. The lfn parameter is required on all ROUTE statements.

lfn	Name of the file to route. lfn can be an input, print, punch, or local file type; it cannot be a primary or direct access file.

The remaining parameters are order independent.

P_i	Description
DC=xx	Disposition code; assumes any one of the following two-character codes (determination of the default code is described following the code definitions).

xx	Meaning	
IN	Release file to input queue. Normal job input file format is required. If the job statement within the file is in error, the file is not released and remains a local file. ROUTE issues a dayfile message explaining the error.	
NO	Release file to input queue. Output not explicitly routed by the job is discarded at job completion.	
LP	Print on any printer.	Print codes
PR	Same as LP.	
LR	Print on 580-12 printer.	
LS	Print on 580-16 printer.	
LT	Print on 580-20 printer.	
SB	Punch system binary.	Punch codes
PB	Same as SB.	
P8	Punch 80-column binary.	
PU	Punch coded.	
PH	Same as PU.	
PL	Plotter.	
SC	Rescind prior routing and change the file type to local. If no prior routing exists, SC is ignored.	

If the DC parameter is omitted and lfn is a deferred routed file (refer to the DEF parameter), the disposition code previously specified remains in effect. If the DC parameter is omitted and lfn is not a deferred routed file, the file name may determine the default.

If DC is omitted and lfn is:	ROUTE assumes DC is:
OUTPUT	DC=LP
PUNCH	DC=PU
PUNCHB	DC=SB
P8	DC=P8

If the DC parameter is omitted, and if the file is not a deferred routed file and does not have one of the preceding file names, the file type determines the default.

p_i	Description

If DC is omitted and the file type is:	ROUTE assumes DC is:
Print	DC=LP
Punch	DC=PH
Any other	DC=SC

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

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

For print files, xx can be the following.

- A4 — Provided for NOS/BE compatibility. If a NOS user specifies A4, the system uses the appropriate EC default (refer to following note).
- A6 — ASCII graphic 63/64-character set.
- A9 — ASCII graphic 95-character set. File lfn must be a 12-bit ASCII file. Refer to the FCOPY control statement earlier in this section.
- B4 — Provided for NOS/BE compatibility. If a NOS user specifies B4, the system uses the appropriate EC default (refer to following note).
- B6 — CDC graphic 63/64-character set.

For punch files, xx can be the following.

- ASCII — Punch ASCII.
- O26 or 026 — Punch O26 mode.
- O29 or 029 — Punch O29 mode.
- SB — Punch system binary.
- 80COL — Punch 80-column binary.

NOTE

If an invalid external characteristic is specified, the queue file processor cannot output the file. The user must not specify a print file characteristic for a punch file or a punch file characteristic for a print file. He also must not specify an external characteristic not available at the site. If EC is not specified, an appropriate EC default is set on the basis of the DC parameter setting and installation options.

p_i	Description
FC=xx	Forms code; specifies routing to the output device that the system operator assigned the forms code xx. This parameter prevents output of a file before its special forms are placed in the output device. xx can be any 2 alphanumeric characters, but the combinations null, AA, AB, AC, AD, AE, and AF give maximum system efficiency. A value of null results when no FC parameter is specified.
FID=xx	NOS/BE parameter included for compatibility. It produces an informative message under NOS.
FM	Implicit remote routing (refer to the note following the last parameter description).
FM=xx	One- to seven-alphanumeric-character family name; indicates routing to a remote batch terminal logged in with the specified family name. The note following the last parameter description describes the default procedures.
IC=xx	Internal characteristics; specifies one of the following: **DIS** Display code. **ASCII** ASCII code. **BIN** Binary. This parameter is normally not specified since its default is automatically established through the disposition code DC.
ID=xx	Selects local device ID from 0 to 67_8 (octal default). This is identical to the ID specified by the SETID control statement.
ID	Implicit central site routing (refer to the note following the last parameter description).
PRI=xx	File priority. This is a NOS/BE parameter included for compatibility. It produces an informative message under NOS.
REP=xx	Number of additional file copies to be routed to a destination.† The range for xx is from 0 to 31; therefore, the number of copies that can be sent ranges from 1 to 32. Values for xx beyond its range are set to zero, an informative message is sent, and one copy is routed to the destination. The default value is zero; only one copy is routed.
SC=xx	Spacing code specifying a programmable format control (PFC) array for the 580 PFC printer. The system is released with two PFC arrays, a default (SC=0), and an alternate (SC=1). The installation can define other PFC arrays. The user can enter any spacing code from 0 to 77_8 (octal default), but if an array is not defined for that code, the default array (SC=0) is used. For more information on spacing codes, refer to the NOS System Maintenance Reference Manual.

†The REP parameter is ineffective in remote batch jobs printed via Export/Import. Regardless of the xx value specified, only one copy is printed at the remote batch site. The REP parameter is effective in remote batch jobs printed via RBF.

p_i	Description
ST=xx	Station ID. This is a NOS/BE parameter included for compatibility. It produces an informative message under NOS.
TID	Implicit remote routing (refer to the note following the last parameter description).
TIC=C	Central site routing. This is a NOS/BE parameter included for compatibility. Its action is identical to the ID parameter.
TID=xx	Terminal ID. This form of the TID parameter is included for NOS/BE compatibility. Under NOS, it is processed the same as TID; however, an informative message is issued stating that xx is ignored.
UN	Implicit remote routing (refer to the note following the last parameter description).
UN=xx	Specifies the user number of the remote batch user to whom the named file is routed. The parameter xx is valid only if it matches the user number of the user performing the route. The matching is character for character, except for those positions containing an * (refer to the note following the last parameter description).

NOTE

For remote batch origin (EIOT) jobs, the following action is taken.

- Specifying an ID, ID=xx, or TID=C parameter causes routing to the central site.
- Specifying or omitting an FM, TID, or UN parameter causes routing to the originating remote batch terminal.
- Specifying an FM or UN parameter with legal arguments causes routing to the specified remote batch terminal.

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

- Specifying an ID, ID=xx, or TID=C parameter causes routing to the central site.
- Specifying a UN, TID, or FM parameter causes routing to the terminal specified by the job's FM and UN at the time of the processing of the ROUTE statement.
- Specifying a UN or FM with legal arguments causes routing to the specified remote batch terminal.

If a job is routed to the input queue with an illegal USER control statement, the following message is issued.

```
DSP - ILLEGAL USER CARD.
SYSTEM ABORT.
```

and the job is aborted with no error exit processing or if submitted from a terminal, the terminal is logged off. The security count for the user number that did the ROUTE is decremented accordingly.

SETID STATEMENT†

The SETID control statement assigns a new identification code for the specified file.

The control statement format is:

SETID(lfn_1=x_1,lfn_2=x_2,...,lfn_n=x_n)

lfn_i	Logical file name.
x_i	New identification code for the file (0 through 67_8). This code must match the device identification code specified in the EST. (The installation establishes the device identification codes.)

The identification code allows the user to route his file to an output device or device group with the same identification code. This is useful when a print file requires special forms.

The file lfn_i must be an input (INFT), local (LOFT), print (PRFT), or punch (PHFT) type file, or the following message is issued.

```
ILLEGAL FILE TYPE
```

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.

†The ROUTE control statement should be used to perform this operation, because the SETID statement will not be supported in future versions of NOS.

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 m is omitted, the system assumes the file is in binary mode. If coded mode is set on an SI tape, the system aborts the job.

If an EOI is encountered before n files are bypassed, file lfn remains positioned at the EOI.

The SKIPF statement has no effect on a primary file since the file is rewound before every operation.

SKIPFB STATEMENT

The SKIPFB control statement directs the system to bypass, in the reverse direction, the specified number of files from the current position of the named file.

The control statement format is:

SKIPFB(lfn,n,m)

lfn	Name of the file to be positioned.
n	Number (decimal) of files to be skipped; if the parameter is omitted, the system assumes n=1.
m	File mode: C for coded, B for binary. If m is omitted, the system assumes the file is in binary mode. If coded mode is set on an SI tape, the system aborts the job.

The system does not backspace past the beginning-of-information (BOI) or load point (tape file) in the event that BOI or load point is encountered before n files are bypassed.

The SKIPFB statement has no effect on a primary file since the file is rewound before every operation.

SKIPR STATEMENT

The SKIPR control statement directs the system to bypass, in a forward direction, the specified number of logical record or file marks 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 record and file marks to be skipped; if this parameter is omitted, the system assumes n=1.
	Level number; $0 \leq \ell \leq 17$. If $0 \leq \ell \leq 16$, both EOR and EOF marks are counted. If $\ell = 17$, only EOF marks are counted.
m	File mode: C for coded, B for binary. If m is omitted, the system assumes the file is in binary mode. If coded mode is set on an SI tape, the system aborts the job.

Consecutive EOR and/or EOF marks are counted separately. If the EOI is encountered before n EOR and EOF marks are bypassed, file lfn remains positioned at the EOI.

The SKIPR statement has no effect on a primary file since the file is rewound before every operation.

SORT STATEMENT

The SORT control statement enables the user to sort a file of line images or statements in numerical order based on leading line numbers consisting of a specified number of digits.

The control statement format is:

SORT(lfn,NC=n)

lfn	Logical file name of the file to be sorted; lfn may be a local file or a direct access permanent file.
n	Number of leading line number digits on which the file is to be sorted; $n \leq 10$. If the NC parameter is omitted, the system assumes n=5.

In the case of duplicate line numbers, all lines other than the first are considered correction lines. All lines with the same number are deleted from the file except the last line encountered.

For input from a time-sharing terminal, SORT deletes a line image or statement if a line number is followed by an empty line or a line number is followed by a blank and a carriage return.

For batch input, SORT deletes a statement or line image if a card containing only the line number is submitted.

If a line number contains more than n digits, the user can delete the line either by entering the first n digits of the line number and pressing the carriage return (terminal input) or by submitting a card containing only the first n digits of the line number (batch input).

After the sort, lfn is packed and set at EOI.

TCOPY STATEMENT

The TCOPY control statement copies X (external) format binary tapes or E (line image), B (blocked), or SI (system internal) format coded tapes to mass storage, to an I format tape, or to an SI binary format tape. It also writes E or B format tapes converted from files on mass storage, I format tape, or SI format binary tape. The X binary and E, B, and SI coded tape formats were supported under earlier versions of NOS. Refer to appendix J for the tape formats of B, E, and X format coded tapes. Now, to access data or write data in one of these formats, the tape must be assigned as an S (stranger) format tape (refer to the tape assignment statements in section 10) and the file copied using the TCOPY statement.

The parameters on the TCOPY control statement can appear in order dependent format, order independent format, or a combination of both. The completely order dependent format is:

TCOPY(lfn_1,lfn_2,format,tc,copycnt,charcnt,erlimit,p_1p_2,lfn_3)

The completely order independent format is:

TCOPY(I=lfn_1,O=lfn_2,F=format,TC=tc,N=copycnt,CC=charcnt,EL=erlimit,
PO=p_1p_2,L=lfn_3)

If order dependent and order independent parameters are mixed in one TCOPY statement, the order dependent parameters must appear in their proper position. All parameters are optional. However, the specification of certain parameters precludes the application of others. A nonapplicable parameter may be ignored or it may be illegal. This is stated in the individual descriptions of the parameters.

The parameters are defined as follows:

Parameter	Description	Default
I=lfn_1	Name of the file to copy from.	INPUT
O=lfn_2	Name of the file to copy to.	OUTPUT
F=format	Data format that specifies the type of conversion for the copy operation. This can be any one of the following.	X

format	Conversion
E	Copy an E format tape to mass storage, an I, or an SI binary tape file, or generate a new E format tape from mass storage, an I, or an SI binary tape file. The E tape must be unlabeled and assigned as S format.
B	Copy a B format tape to mass storage, an I, or an SI binary tape file, or generate a new B tape from mass storage, an I, or an SI binary tape file. The B tape must be unlabeled and assigned as S format.

Parameter	Description		Default
	format	Conversion	
	X	Copy an X format tape to mass storage, an I, or an SI binary tape file. The unlabeled input tape must be assigned an S format, with noise size of eight frames for seven-track tape or six frames for nine-track tape (refer to NS parameter on tape assignment control statement).	
	SI	Copy an SI coded format tape to mass storage, an I, or an SI binary tape file. The labeled or unlabeled input tape must be assigned as S format, with noise size of eight frames for seven-track tape or six frames for nine-track tape (refer to NS parameter on tape assignment control statement). If the SI coded input tape is completed before EOI is encountered; the position of the input tape after the copy is indeterminate. This is because control words are used on the SI coded tape read via S format (EOF on an SI coded tape is a level 17_8 block terminator, whereas EOF on an S tape is a tape mark).	
TC=tc	Specifies the copy termination condition used in conjunction with N=copycnt. The termination condition can be specified as follows:		Copy to double EOF (TC=D or TC=EOD)
	F or EOF	When this TC value is set, the N keyword specifies the number of files to copy.	
	I or EOI	This specifies a copy to the end of information. The N keyword is ignored.	
	D or EOD	When this TC value is set, the N keyword is the number of double EOFs to copy to.	
N=copycnt	Copy count used by the copy termination condition TC.		1
CC=charcnt	The character count which determines maximum block size (line length) in characters for an E or B format tape. This parameter can only be specified on an E or B format tape copy.		136 characters for E format; 150 characters for B format.

Parameter	Description	Default
EL=erlimit	Error limit which sets the number of nonfatal errors allowed before abort. This includes parity errors and block-too-large errors which are returned by the tape subsystem after completing recovery procedures. It also includes illegal block format errors (invalid byte-count and/or unused bit count) for X format and SI coded format tapes. Error limit is ignored when generating an E or B format tape from mass storage, an I format, and an SI binary format file since control word read is not used. Error limit is likewise ignored if the input file device does not support control word read (terminals). In that case, any error aborts the job.	Zero
PO=p_1p_2	One or both of the following processing options (not separated by commas).	
	E Input blocks with parity errors or block-too-large errors are processed (copied).	Error blocks are skipped.
	T When generating a B or E format tape, blocks exceeding the maximum block size (refer to the CC parameter) are truncated. PO=T is illegal for other file conversions.	Lines exceeding the maximum line size are split into multiple blocks.
L=lfn_3	Name of an alternate output file to receive parity error messages when extended error processing is in effect (nonzero EL specified), in which case, the file name lfn_3 must not be the same as lfn_2.	OUTPUT

Example:

The following TCOPY statement combines order dependent and order independent parameters:

```
TCOPY(TAPE1,FILE2,E,CC=200,EL=12)
```

The input file TAPE1 is an E format tape (assigned as an S format tape). It has a maximum of 200 characters per line. The copy terminates when a double EOF is encountered (default). The output file FILE2 can be a mass storage file or an I or SI binary format tape. The error limit allows up to 12 nonfatal errors (parity/block-too-large), and the bad data is skipped (default) with informative error messages written to the file OUTPUT (default).

The TCOPY statement begins a copy operation at the current position of both files and continues until the copy termination condition is met or EOI is encountered. This termination condition can be a file count, double EOF count, or EOI. If the copy is terminated by a double EOF (for TC=EOD parameter), the second EOF is detected on lfn_1 but is not transferred to lfn_2. If lfn_1=lfn_2, the named file is read until the termination condition is satisfied or EOI is encountered. An SI coded tape can be positioned correctly only to EOI (refer to the F=SI parameter description).

If a copy specifies a file count TC=EOF, and EOI is encountered on the input file before the file count is satisfied, an additional EOF is written on the output file only if data or records have been transferred since the previous EOF was written (or since the beginning of the copy, if no EOFs have been encountered).

The EL or PO=E options provide extended error processing. This allows the processing or skipping of blocks with parity errors or block-too-large errors. If EL is set to a value greater than zero, a parity error or block-too-large error on the input tape generates the following message on the alternate output file.

PARITY/BLOCK TOO LARGE ERROR IN BLOCK n.

n is the decimal block count of the block in error. The block count for the first block to be copied is initially set to zero and is incremented by one for every block and every EOF processed. For X and SI coded formats, an illegal block format error (illegal byte count and/or unused bit count) produces the following message on the alternate output file.

ILLEGAL FORMAT IN BLOCK n.

When creating a B format tape from a mass storage, I, or SI binary format file, a block shorter than the noise size specified on the tape assignment statement is blank filled to the noise size. A noise block could also be generated when a block exceeding the maximum block size for the B format tape is split into multiple blocks. If the PO=T parameter is specified, blocks exceeding the maximum block size are truncated.

When creating an E format tape from a mass storage, I, or SI binary format file, blocks that exceed the maximum block size for the E format tape are split into multiple blocks. If a continuation block contains only the end-of-line indicator (zero word), the continuation block is discarded. If the PO=T parameter is specified, blocks exceeding the maximum block size are truncated (all continuation blocks are discarded).

TDUMP STATEMENT

The TDUMP control statement lists a file in octal and/or alphanumeric format. It dumps the entire file or the specified number of lines, records, or files. If more than one limit is set, the limit reached first overrides the others.

NOTE

TDUMP produces unpredictable results when dumping an S, L, or F format tape file. The user should use the COPY statement to convert the S, L, or F format tape file to a mass storage file or to an I or SI binary format tape file before attempting to dump the file using TDUMP.

The control statement format is:

TDUMP(p_1,p_2,...,p_n)

p_i Any of the following in any order:

I=lfn_1 One to seven alphanumeric characters naming the local file to be dumped (default is TAPE1).

L=lfn_2 One to seven alphanumeric characters naming the local file to which the output is written (default is OUTPUT). If lfn_2 is not a local file, TDUMP creates it. It does not rewind lfn_2 before or after the dump.

O — Octal dump only.

A — Alphanumeric dump only.

If both O and A are specified, the last one overrides. If neither O nor A is specified, TDUMP lists both an octal and an alphanumeric dump.

R=rcount — Maximum decimal number of records to be dumped. If R is omitted or set to zero, the dump continues to EOI.

NOTE

The record count restarts at each EOF.

F=fcount — Maximum decimal number of files to be dumped. If F is omitted, the dump continues to EOI. If F=0, dump continues until an empty file (double EOF) or EOI is encountered.

N=lines — Maximum decimal number of lines to be dumped. If N is omitted or set to zero, the dump continues to EOI. The blank line output with the end-of-record, end-of-file, end-of-information, and ABOVE LINE REPEATED messages is included in the line count.

NR — Do not rewind file lfn_1 before dump (default is to rewind lfn_1).

Example:

Two lines, each containing the alphabet, were input to file X from a time-sharing terminal. File X was dumped to file Y producing the following output.

```
      - FILE DUMP -        TDUMP,I=X,L=Y.                                                    78/10/23. 08.05.51. PAGE    1.

F  1 R   1 W     0- 0102 0304 0506 0710 1112   1314 1516 1720 2122 2324   2526 2730 3132 0000 0000   0102 0304 0506 0710 1112
                    A B  C D  E F  G H  I J     K L  M N  O P  Q R  S T     U V  W X  Y Z              A B  C D  E F  G H  I J
F  1 R   1 W     4- 1314 1516 1720 2122 2324   2526 2730 3132 0000 0000
                    K L  M N  O P  Q R  S T     U V  W X  Y Z
-- END OF RECORD --

-- END OF INFORMATION --

-- END OF DUMP --
```

The prefix

F 1 R 1 W 0

means file 1, record 1, word 0. The zeros following each alphabet are the end of line indicators.

UNLOAD STATEMENT

The UNLOAD control statement releases files assigned to the job and may release file space (depending on the file type).

The control statement formats are:

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

or

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

The first format unloads the named files (lfn_1,lfn_2,...,lfn_n). The second format unloads all files assigned to the job except the named files. If no files are named on the second format, the asterisk specification unloads all files assigned to the job. In a CCL procedure the second format does not release the CCL work files (ZZZZZC0, ZZZZZC1, and ZZZZZC2). An UNLOAD control statement not in a CCL procedure will release these files.

The UNLOAD statement performs the same function as the RETURN control statement except as noted below. Refer to the description of the RETURN statement given earlier in this section to determine the operation performed for each file type.

If the file being unloaded is a magnetic tape file or a direct access file residing on an auxiliary removable pack and the job requires more than one tape or pack concurrently, the UNLOAD statement differs from the RETURN statement in the following ways.

- It does not decrease the number of tapes or packs scheduled for the job with the RESOURC control statement.
- It causes the following operations to be performed for magnetic tape files, if the previous operation was a write.
  - If the tape is ANSI labeled, the system writes a tape mark, an EOF1 label, and three tape marks and then unloads the tape.
  - If the tape is unlabeled and the data format specified on the ASSIGN, LABEL, or REQUEST card is S, L, or F, the system writes four tape marks and then unloads the tape.
  - If the tape is unlabeled and the data format is I or SI, the system writes a tape mark, an EOF1 label, and three tape marks and then unloads the tape.

Refer to Magnetic Tape Files in section 2 and Tape Management control statements in section 10 for further information about tape files, and to appendix G for a description of an EOF1 label.

UNLOCK STATEMENT

The UNLOCK control statement rescinds the LOCK command and clears the write interlock bit for the specified file.

The control statement format is:

UNLOCK(lfn_1,lfn_2,...,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.

Library files cannot be unlocked.

VERIFY STATEMENT

The VERIFY statement performs a binary comparison of all data from the current position of the files specified. The comparison is meaningful if the files are within the range of compatible formats listed in table 1-7-2.

The control statement format is:

VERIFY(lfn_1,lfn_2,p_1,p_2,...,p_n)

lfn_1 Name of the first file; if this parameter is omitted, TAPE1 is assumed.

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

p_i Any of the following in any order:

N=0	Verification terminates on the first empty file encountered on either file.
N=x	Verify x files; default is N=1.
N	Verification terminates when end of information is encountered on both files.
E=y	List the first y errors encountered on the comparison. If E is omitted, the system assumes E=100.
E	Same as E=0, no errors are listed.
L=lfn_3	List errors on file lfn_3. If L is omitted, the system assumes L=OUTPUT.
A	Abort after verification completed if errors occurred.
R	Rewind both files before and after the verification.
C	Coded file mode is set on both files. This is applicable only to S and L format tapes. If coded mode is set on an SI tape, the system aborts the job.
C1	Coded file mode is set on the first file only. This is applicable only to S and L format tapes. If coded mode is set on an SI tape, the system aborts the job.
C2	Coded file mode is set on the second file only. This is applicable only to S and L format tapes. If coded mode is set on an SI tape, the system aborts the job.
BS=bsize	Defines the maximum block size (PRU size) in central memory words for an S or L tape. This parameter is legal only for S and L tape verifies. The default for an S tape is 1000_8 words, and for an L tape, it is 2000_8 words.

Whenever words on lfn_1 and lfn_2 do not match, the VERIFY statement lists the following.

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

If excess records are encountered on lfn_1 or lfn_2, the following message is listed.

n EXCESS RECORD(S) ON lfn.

n is the decimal number of excess records. The title line of the error list file contains the decimal number of the logical file being verified. If a nonstandard file (one in which an EOI or EOF is not preceded by an EOR) is compared with a standard file, the VERIFY statement lists the following message.

r EOR MISSING ON lfn

r	Record number in decimal.
lfn	Name of the nonstandard file.

If EOI is encountered on one input file (lfn_1 or lfn_2) and there are still files remaining on the other input file, each excess file generates the following message.

n RECORD(S) IN EXCESS FILE m ON lfn.

n is the decimal number of excess records in logical file number m.

If errors are encountered, the following warning message is issued to the user's dayfile.

VERIFY ERRORS.

If any pair of lfn_1, lfn_2, and lfn_3 are identical, the following fatal message is issued.

FILE NAME CONFLICT.

If lfn_1 or lfn_2 did not exist prior to the verify, the following warning message is issued.

FILE NOT FOUND - lfn.

In a verification involving S, L, or F format tapes, the VERIFY statement first clears the extraneous data in the last word of each block (as specified by the byte count and the unused bit count) and then makes the comparison. On these formats, every block is considered a record (returns EOR status).

If a verification of an L or F format tape requires additional field length, the VERIFY statement increases the field length as needed. If the field length requirement exceeds the user's maximum field length, the verify is aborted with the error message

VERIFY FL ABOVE USER LIMIT.

The maximum block size for an L format tape is specified by the BS=keyword or its default. The maximum block size for an F format tape is calculated from the frame or character count specified on the control statement when the file is assigned.

Verification is not guaranteed when the logical structures of the two files are incompatible. Before the VERIFY statement makes a comparison of such files, it issues the warning message

FILE STRUCTURES NOT COMPATIBLE.

TABLE 1-7-2. COMPATIBLE FILE STRUCTURES FOR THE VERIFY STATEMENT

FIRST FILE (lfn_1)		SECOND FILE (lfn_2)					
			Tape Formats				
		Mass Storage	I	SI	S	L	F
Mass Storage		Yes	Yes	Yes	No	No	No
Tape Format	I	Yes	Yes	Yes	No	No	No
	SI	Yes	Yes	Yes	No	No	No
	S	No	No	No	Yes	No	No
	L	No	No	No	No	Yes	No
	F	No	No	No	No	No	Yes

NOTE

The No entries indicate that the logical structures of the files compared are incompatible. The VERIFY statement may accept those combinations, but the results require the user to make a knowledgeable correlation of results with the format descriptions in section 10. In some cases, the verification of an incompatible pair may result in a VERIFY GOOD message; otherwise, a VERIFY ERRORS message is listed.

WRITEF STATEMENT

The WRITEF control statement directs the system to write a specified number of file marks on the named file.

The control statement format is:

WRITE(lfn,x)

lfn — Name of the file to be written on.

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

If the last operation to the file was a write that did not end with the writing of an EOR or EOF, WRITEF writes a record mark before it writes the specified number of file marks. For all other cases, WRITEF writes the file marks without a preceding record mark.

WRITER STATEMENT

The WRITER control statement directs the system to write a specified number of empty records on the named file.

The control statement format is:

WRITER(lfn,x)

lfn — Name of the file to receive the empty records.

x — Number of empty records to be written; if this parameter is omitted, the system assumes x=1.

PERMANENT FILE CONTROL STATEMENTS 8

With permanent file control statements, the user can create, access, and purge permanent files.† The owner of a permanent file can also control other users' access to the file.

The following are the permanent file control statements and their functions.

Statement	Function
APPEND	Appends data to an indirect access permanent file.
ATTACH	Assigns a direct access permanent file to a job.
CATLIST	Lists permanent file information.
CHANGE	Changes permanent file characteristics.
DEFINE	Creates a direct access permanent file.
GET	Retrieves a copy of an indirect access permanent file as a local file.
OLD	Retrieves a copy of an indirect access permanent file as the primary file.
PACKNAM	Specifies an auxiliary device from which permanent files are to be accessed.
PERMIT	Grants permanent file access permission to other users.
PURGALL	Purges all files having the specified characteristics.
PURGE	Purges the named files.
REPLACE	Copies a local file over an indirect access permanent file.
SAVE	Creates an indirect access permanent file.

If an error occurs in an operation on one file of a multifile request, the operation is not performed on the subsequent files. For example, if an error occurs in the processing of file B on the following control statement:

```
GET(A,B,C,D)
```

files C and D are not processed.

For more information on permanent files, refer to Permanent Files, Mass Storage Files, and Mass Storage File Residence in section 2.

To determine the meaning of a permanent file error message, refer to appendix B.

†The batch user cannot access permanent files unless he has included a USER statement in his job file.

COMMON CONTROL STATEMENT PARAMETERS

Some permanent file control statements can process several files. Each file is identified by a one- to seven-character file name. The permanent file operation is performed on each file in the sequence specified on the statement. If the NA parameter is not specified on the statement, and if an error occurs while one of the files is being processed, the job step terminates.

When this happens, the files preceding the file in error are processed, but the files succeeding it are not. If the NA parameter is specified on the statement, the job step does not terminate as a result of the error, and all files are processed except the file in error.

A slash separates the file names from other parameters. The parameters specified after the slash are optional and order independent. Parameters specified after the slash affect all files named before the slash.

Permanent file access is controlled by the access category (CT), user number (UN), file access password (PW), and access mode (M) parameters. Permanent file residence is determined by the preferred residence (PR), auxiliary device (PN), and device type (R) parameters. Processing options include the no abort (NA), subsystem selection (SS), and backup copy (BR) parameters.

In the following list, parameters common to more than one permanent file control statement are described in alphabetic order. For the format of each control statement, refer to the control statement descriptions following the parameter descriptions.

Parameter	Description
BR=br	Backup copy requirement. The user can request that a tape backup copy, an MSF backup copy, or no backup copy be kept of the permanent file. If BR=br is omitted when the file is created, a backup copy of the file is stored on tape. For more information, refer to Mass Storage Facility in section 2.

br	Backup Requirement
Y	A tape backup copy should be kept, even if a copy of the file exists on MSF.
MD	A tape backup copy need not be kept, if a copy of the file exists on MSF.
N	A backup copy need not be kept for this file.

CT=ct — File access category. If CT=ct is omitted when the file is created, the file is private. The file category can be changed later.

ct	Category
P, PR, or PRIVATE	Private files; available for access only by their creator and by those granted explicit access permission by the file creator (refer to the PERMIT control statement).
S or SPRIV	Semiprivate files; available for access by a user who knows the file name, user number, and password and who has not been explicitly denied permission to the file (M=NULL parameter on a PERMIT statement).

Parameter	Description
	ct — Category: The system records the user number of each user who accessed the file, the number of accesses made, and the date and time of the last access by each user. The file creator can list this information with the CATLIST(LO=FP) statement.
	PU or PUBLIC — Public files; 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 and the date and time of the last access, but does not record user numbers.
NA	No abort. Normally, an error in statement processing terminates the job. However, if NA is specified, the job does not terminate as a result of the error. The error is handled in one of two ways. • If the error condition is temporary, job processing is suspended until the error condition ends (for example, when a requested direct access file is busy or a requested auxiliary device is not available). In a multimainframe environment, job processing is suspended until the error condition ends and the rollout delay time has elapsed. • If the error condition is not temporary, the job continues with the next operation. The job processes the next file listed on the statement or, if no more files are listed, processes the next control statement.
PN=packnam	One- to seven-character pack name used in conjunction with the R parameter 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. 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 concurrently uses two or more auxiliary disk packs or an auxiliary pack and a magnetic tape.
PR=pr	Preferred residence. If PR=pr is omitted when a direct access file is defined, no preference is assumed. For more information, refer to Mass Storage Facility in section 2. pr — Preference M — The user prefers that the direct access file be copied to MSF between accesses. However, the system might not copy the file to MSF despite the specified preference. N — The user has no preference.
PW=passwrd	One-to-seven-character file password. If a password is assigned to the file, users other than the file creator must specify the password when accessing the file.

Parameter	Description
	If only the keyword PW is specified, the user must include the password as a single-line record in the INPUT file. In a time-sharing job, the password is requested by a ? prompt at the terminal.
R=r	Device type on which the file resides or is to reside.

r	Device
DE	Extended memory†
DIi	844-21 Disk Storage Subsystem ($1 \le i \le 8$) (half-track)
DJi	844-41 or 844-44 Disk Storage Subsystem ($1 \le i \le 8$) (half-track)
DKi	844-21 Disk Storage Subsystem ($1 \le i \le 8$) (full-track)
DLi	844-41 or 844-44 Disk Storage Subsystem ($1 \le i \le 8$) (full-track)
DMi	885 Disk Storage Subsystem ($1 \le i \le 3$) (half-track)
DQi	885 Disk Storage Subsystem ($1 \le i \le 3$) (full-track)
DP	Distributive data path to ECS†

The R parameter can be specified on any permanent file control statement to identify (with the PN parameter or a previous PACKNAM control statement) the auxiliary device on which the permanent file resides. R is specified only if the installation defines the auxiliary device as removable, and if the desired device type differs from the installation-defined default device type. If the R parameter is specified without the PN parameter in a control statement, R is ignored on all control statements except the DEFINE statement (refer to the DEFINE control statement in this section).

SS=subsystem

Specifies the time-sharing subsystem to be associated with the file. The subsystem can be specified on a SAVE or CHANGE statement with one of the following entries or its abbreviation (the abbreviation is the first three characters of the entry).

Subsystem	Time-Sharing Subsystem
BASIC	BASIC
BATCH	Batch
EXECUTE	Execute
FORTRAN	FORTRAN 5
FTNTS	FORTRAN Extended 4
NULL	Null

If the SS parameter is specified without a subsystem, the currently active subsystem is associated with the file. If the SS parameter is omitted, no subsystem (null) is associated with the file, unless the file is the primary file. In that case, the currently active subsystem is associated with the file.

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

Parameter	Description
UN=usernum	User number. This parameter must be specified if the requested permanent file is in another user's catalog. To access the file, the user must have one of the following permissions. • Explicit permission. The owner of the file granted access permission to the user with a PERMIT statement. • Implicit permission. The file is semiprivate or public and so can be accessed by users who know the file name, user number, and password and who have not been explicitly denied permission. • Automatic permission. A user has automatic permission to access files in the catalog of another user if his user number contains asterisks, and if all nonasterisk characters match characters in the other user's user number. The UN keyword establishes alternate access validation, even if the specified user number is the one under which the job is currently being run. (Refer to the explanation of secondary user numbers in USER Statement in section 6.)

APPEND STATEMENT

The APPEND control statement adds information to the end of an existing indirect access file without retrieving the file. The APPEND statement cannot append data to direct access files.

The control statement format is:

APPEND(pfn,lfn_1,lfn_2,...,lfn_n/UN=usernum,PW=passwrd,PN=packnam,R=r,NA)

pfn — Name of the indirect access permanent file to which data is appended. The user does not retrieve the indirect access file before the append operation, but he must have permission to access the file in append, modify, or write mode.

If a local copy of the indirect access file is assigned to the job, the append operation does not change the local copy.

lfn_i — Name of a local mass storage file to be appended to pfn. The file must be assigned to the job.

The full descriptions of the following optional parameters are given at the beginning of this section.

Parameter	Description
UN=usernum	User number. Specified if pfn is in another user's catalog.
PW=passwrd	File password. Specified if UN=usernum is specified and pfn has a password.
PN=packnam	Auxiliary device name. Specified if pfn resides on an auxiliary device.

Parameter	Description
R=r	Device type. Specified if an auxiliary device on a device type other than the installation-defined default is to be used.
NA	No abort option. If NA is specified, processing errors do not terminate the job.

The logical structure of the files is retained; that is, EORs and EOFs are appended as well as data. The append operation removes the EOI mark at the end of file pfn, but keeps all EOR and EOF marks.

ATTACH STATEMENT

The ATTACH control statement assigns a direct access permanent file to a job.

The control statement format is:

ATTACH(lfn_1=pfn_1,lfn_2=pfn_2,...,lfn_n=pfn_n/M=m,UN=usernum,PW=passwrd,
PN=packnam,R=r,NA,RT)

lfn_i=pfn_i One- to seven-character file name lfn_i references direct access file pfn_i while the file is assigned to the job. If lfn_i= is omitted, the direct access file is referenced by its permanent file name, pfn_i. User access is directly to the permanent file; no working copy is generated.

If a local file name lfn_i exists when this statement is processed, it is discarded (even if statement processing does not complete due to an error).

The full descriptions of the following optional parameters (except M=m and RT) are given at the beginning of this section.

M=m File access mode requested. If M=m is omitted, M=READ is assumed.

If m is:	The user can:	Concurrently, other users can:
E or EXECUTE (execute mode)	Execute the file.	Read or execute the file.
R or READ (read mode)	Read or execute the file.	Read or execute the file.
RA or READAP (read append mode)	Read or execute the file.	Read or execute the file; one user can lengthen the file.
RM or READMD (read modify mode)	Read or execute the file.	Read or execute the file; one user can lengthen or rewrite the file.
A or APPEND (append mode)	Read, execute, or lengthen the file.	Read or execute the file.

If m is:	The user can:	Concurrently, other users can:
M or MODIFY (modify mode)	Read, execute, lengthen, or rewrite the file.	Read or execute the file.
W or WRITE (write mode)	Read, execute, lengthen, rewrite, or shorten the file.	No access allowed.

The file cannot be changed by the user (including the creator) unless the file is attached in append, modify, or write mode. Refer to table 1-8-1 for the access mode granted when the file is already attached by another user.

If the file is attached in append, modify, or write mode, the current date is recorded as the last modification date even if the file is not altered.

UN=usernum User number. Specified if the permanent file(s) is in another user's catalog.

PW=passwrd File password. Specified if UN=usernum is specified, and if the permanent file has a password.

PN=packnam Auxiliary device name. Specified if the permanent file(s) resides on an auxiliary device.

R=r Device type. Specified if an auxiliary device on a device type other than the installation-defined default is used.

NA No abort option. If NA is specified, processing errors do not terminate the job. The NA parameter allows the user to attach a direct access file that has an error status entry in its catalog (refer to the LO=0 parameter description in CATLIST Statement in this section).

RT Real-time processing. If RT is specified, the job continues processing after requesting staging of a direct access file from MSF to disk. If staging is not required (the file is already resident on disk), the file is assigned to the job. If the file is staged, the user must issue a second ATTACH statement to assign the file to the job.

If RT is omitted and the file resides only on the MSF, job processing is suspended while the MSF file is staged to disk and assigned to the job. For more information on the MSF, refer to Mass Storage Facility in section 2.

If an auxiliary device has been previously specified by a PACKNAM statement, the system attempts to find pfn_i on the auxiliary device rather than on a family device.

In the first column of table 1-8-1 is the access mode in which the first user attached a direct access file. The remaining columns are the access modes that may be requested by another user. At the intersection of the row and column is the access mode granted to the other user (assuming he has the appropriate access permission). Busy means that the other user is sent a message that the file is busy and the job step is aborted unless NA is specified.

Assuming that no user has the file attached in append, modify, or write mode, 12 285 users can attach the file concurrently (4095 in read mode, 4095 in read append mode, and 4095 in read modify mode). If a user has the file attached in append mode, 8190 other users can attach the file (4095 is read append mode and 4095 in read modify mode). If a user has the file attached in modify mode, 4095 other users can attach it in read modify mode. If a user has the file attached in write mode, no other user can attach the file.

TABLE 1-8-1. ACCESS MODE GRANTED WHEN ATTACHING A CURRENTLY ATTACHED DIRECT ACCESS FILE

Mode in Which First User Attached File	Access Mode Requested by Another User						
	Write	Modify	Append	Read	Read Modify	Read Append	Execute
Write	Busy	Busy	Busy	Busy	Busy	Busy	Busy
Modify	Busy	Busy	Busy	Busy	Read modify	Busy	Busy
Append	Busy	Busy	Busy	Busy	Read modify	Read append	Busy
Read	Busy	Busy	Busy	Read	Read modify	Read append	Execute
Read Modify	Busy	Modify	Append	Read	Read modify	Read append	Execute
Read Append	Busy	Busy	Append	Read	Read modify	Read append	Execute
Execute	Busy	Busy	Busy	Read	Read modify	Read append	Execute

CATLIST STATEMENT

The CATLIST control statement lists information about the user's permanent files or the permanent files the user can access in the catalogs of other users.

The control statement format is:

CATLIST(LO=p,FN=pfn,UN=usernum,PN=packnam,R=r,DN=dn,NA,L=lfn)

LO=p One of the following list options. If LO is omitted, LO=0 is assumed.

p	Description
F	Lists pertinent information about each file in the user's catalog. An example of this list option is given following the parameter descriptions. If another user number is specified (UN=usernum), the user receives a listing of all the files he can access in the other user's catalog. The passwords for files in another user's catalog are not listed. Passwords must be obtained from the user.
FP	Lists the access permissions granted for the file specified on the FN=pfn parameter. If a user number is also specified (UN=usernum), only the file permission granted to that user is listed. The user numbers listed include those that have been granted explicit permission to access the file (private files only) and those that have accessed the file through implicit permission (semiprivate files only). (User numbers are not recorded for accesses to public files.) An asterisk follows the user number/permission mode if explicit permission has been granted this user.

p	Description
0 (zero)	Lists alphabetically the names of the indirect access files and direct access files in the user's catalog. If he specifies a user number (UN=usernum), the user receives a list of the files he can access in the other user's catalog. An asterisk preceding a file name indicates an error status is set in the catalog entry for the file. The cause of the error may be one of the following. • EOI was altered during mass storage recovery. • Error exists in file data and/or permit entries. To clear an error status flag, refer to CHANGE Statement in this section.
P	Lists only the user numbers of users who have access to the specified private file or who have accessed the specified semiprivate file. This option requires that a file name be specified (FN=pfn).

FN=pfn Permanent file name. This parameter is required when listing access permissions granted (when LO=FP or LO=P is specified).

If FN=pfn and LO=0 are specified, a message informs the user whether or not the file is found in the user's catalog. If FN=pfn, LO=P, and UN=usernum are specified, a message informs the user whether or not permission to access that file has been granted to that user number (usernum FOUND.).

If pfn contains one or more asterisks, the CATLIST statement lists catalog information for the subset of files whose names match except where the asterisks appear. For example, FN=***OPL lists all six-character file names ending in OPL. The asterisk is invalid when listing permit information with the LO=FP and LO=P list options.

UN=usernum User number. This parameter has two purposes.

- For list options LO=F and LO=0, it specifies the alternate catalog for which the user desires catalog information.
- For list options LO=FP and LO=P, it requests the permission information recorded for the specified alternate user.

When the short list options, LO=0 and LO=P, are specified, a message informs the user whether or not the file or user number has been found.

PN=packnam Auxiliary device name. This parameter specifies that CATLIST should use the catalog information on the named auxiliary device.

R=r Device type on which the permanent file catalog resides. R=r is used with the PN and NA parameters (refer to the R parameter description at the beginning of this section).

DN=dn Device number assigned to a device at system initialization. If specified, the CATLIST statement lists catalog information from that device.

NA — No abort option. If NA is specified, processing errors do not terminate the job.

L=lfn — Output file name. lfn is the name of the file assigned to the job on which CATLIST information is written. If L=lfn is omitted, the system assumes L=OUTPUT.

lfn is not rewound before or after the CATLIST operation.

Example 1:

Listing of current files in the catalog of BJK2201. The statement is entered by user BJK2201 in the form: CATLIST (it is not necessary to specify the LO=0 option since it is the default value).

```
 CATALOG OF BJK2201            FM/NOSCLSH 79/03/14. 09.10.47.

 INDIRECT ACCESS FILE(S)

  ADD        EXAM       GRADES     ID         MODIFY2    RESEQ      XX
  CAPITAL    FIND      *HEROFTN    LIST       PRIME      T

 DIRECT ACCESS FILE(S)

  DIRFILE    DRFILE     TV

        13 INDIRECT ACCESS FILE(S),    TOTAL PRUS =         14.
         3 DIRECT ACCESS FILE(S),      TOTAL PRUS =          2.
```

An asterisk preceding a file name indicates error flag set.

Example 2:

Listing of current files that begin with the letters PROC in the catalogue of WAC3651. The statement is entered by user WAC3651 in the form: CATLIST,FN=PROC***.

```
  CATALOG OF WAC3651             FM/NOSCLSH 80/04/18. 13.27.02.

 INDIRECT ACCESS FILE(S)

  PROCART    PROCFIL    PROC1      PROC1A

         4 INDIRECT ACCESS FILE(S),    TOTAL PRUS =          15.
```

Example 3:

Listing of alternate users that have accessed file PRIME in the catalog of BJK2201. The statement is entered by user BJK2201 in the form: CATLIST,LO=P,FN=PRIME.

```
 CATALOG OF BJK2201            FM/NOSCLSH 79/03/08. 07.48.55.
         FILE NAME PRIME
  USER NUMBER(S)

  CML2011    JLC2016    KXK4277

         3 USER(S)
```

Example 4:

Listing of current files in the catalogue of SAH3523. The statement is entered by user SAH3523 in the form: CATLIST,LO=F.

```
CATALOG OF SAH3523              FM/NOSCLSH 79/07/20. 13.49.09.  PAGE    1

   FILE NAME ACCESS FILE-TYPE LENGTH DN CREATION  ACCESS    DATA MOD
    PASSWORD MD/CNT INDEX  PERM. SUBSYS DATE/TIME DATE/TIME DATE/TIME
    PR BR RS


   1 FJOB     IND. PRIVATE        21    79/06/08. 79/06/18. 79/06/18.
                22         WRITE        13.31.46. 11.10.23. 11.10.23.
     N MD D

   2 PROGF    DIR. PRIVATE        88  * 79/07/20. 79/07/20. 79/07/20.
                0          WRITE        13.40.52. 13.40.52. 13.40.52.
     M MD A

        1 INDIRECT ACCESS FILE(S),    TOTAL PRUS =        21.
        1 DIRECT ACCESS FILE(S),      TOTAL PRUS =        88.
```

The page heading gives the user number, the family name, and the date and time. If the PN=packnam parameter is specified, the family name in the heading is replaced by PN/packnam.

Column headings are printed when the LO=F parameter is specified.

The following are the column headings and their meanings.

Heading	Meaning
FILE NAME	Permanent file name.
ACCESS MD/CNT	ACCESS MD is the permanent file type; direct access file (DIR) or indirect access file (IND). ACCESS CNT is the number of times the file has been accessed. It is listed on the line below ACCESS MD.
FILE-TYPE	File access category. The entry is PRIVATE, SPRIV, or PUBLIC.
LENGTH	Length of the file in decimal PRUs.
DN	Device number of the mass storage device on which the direct access file is stored. If the file resides on the master device, this field contains an asterisk.
PASSWORD	File password. It is not listed if the file belongs to another user.
INDEX	This heading is not used by the CATLIST statement. It is used by the PFCAT system utility.
PERM.	Permission mode allowed the user. The entry is WRITE, MODIFY, APPEND, READ, READMD, READAP, or EXECUTE.

Heading	Meaning
SUBSYS	Time-sharing subsystem associated with the file. The possible entries are FORT., FTNTS, BASIC, EXEC., and BATCH. If this field contains no entry, a subsystem is not associated with the file (null).
CREATION DATE/TIME	File creation time and date in the following format. yy/mm/dd. hh.mm.ss.
ACCESS DATE/TIME	Time and date of the last access to the file.
DATA MOD DATE/TIME	Time and date of the last modification of the file data.
PR	Preferred residence for this file. M means MSF and N means no preference. For more information, refer to the description of the PR parameter at the beginning of this section.
BR	Requested backup copy for this file. Y means tape backup, MD means MSF or tape backup, and N means no backup required. For more information, refer to the description of the BR parameter at the beginning of this section.
RS	Actual residence of the file. D means disk, A means MSF, and B means copies of the file exist on disk and MSF. For more information, refer to Mass Storage Facility in section 2.

CHANGE STATEMENT

The CHANGE control statement changes certain characteristics of a permanent file. The file need not be assigned to the job.

The control statement format is:

CHANGE(nfn_1=ofn_1,nfn_2=ofn_2,...,nfn_n=ofn_n/PW=passwrd,CT=ct,M=m,BR=br,PR=pr,
SS=subsystem,PN=packnam,R=r,NA,CE)

nfn_i=ofn_i	One- to seven-character file name nfn_i replaces old permanent file name ofn_i. If no name change is desired, only ofn_i is specified.

The full descriptions of the following optional parameters (except CE and M=m) are given at the beginning of this section.

PW=passwrd	New password. If PW=0 is specified, the CHANGE statement clears the old password without setting a new password.
CT=ct	New access category for the file; entries are PRIVATE, SPRIV, and PUBLIC.

M=m — New alternate user permission mode for semiprivate and public files. For direct access files, refer to the permission modes described in the DEFINE statement description; for indirect access files, refer to the permission modes described in the SAVE statement description.

BR=br — New backup copy selection; entries are tape (Y), MSF (MD), and no backup (N).

PR=pr — New preferred residence; entries are MSF (M) and no preference (N).

SS=subsystem — New time-sharing subsystem to be associated with the file; entries are BASIC, BATCH, EXECUTE, FORTRAN, FTNTS, and NULL.

PN=packnam — Auxiliary device on which the file resides. This parameter cannot specify a new file residence.

R=r — Device type on which the file resides. This parameter cannot specify a new file residence.

NA — No abort option. If NA is specified, processing errors do not terminate the job.

CE — Clear file error code. For more information, refer to section 5 in volume 2.

DEFINE STATEMENT

The DEFINE control statement can create an empty direct access permanent file. It can also change a file of the local file type into a direct access file, if the file resides on a permanent file device.

The control statement format is:

DEFINE(lfn_1=pfn_1,lfn_2=pfn_2,...,lfn_n=pfn_n/PW=passwrd,CT=ct,M=m,BR=br,PR=pr,
PN=packnam,R=r,S=space,NA)

lfn_i=pfn_i — If the DEFINE statement creates an empty direct access permanent file, lfn_i is specified if the user wants (in the current job) to reference the file by a name other than its permanent file name, pfn_i. Each file name can be from one to seven characters.

If the DEFINE statement defines an existing local file as a direct access file, lfn_i is the name of the local file, and pfn_i is its new permanent file name. If lfn_i= is omitted, pfn_i is assumed to be the local file name and the permanent file name.

The full descriptions of the following optional parameters (except M=m and S=space) are given at the beginning of this section.

PW=passwrd — One- to seven-character password that other users must specify to access the file.

CT=ct — Access category of the defined file; entries are PRIVATE, SPRIV, and PUBLIC. If CT=ct is not specified, CT=PRIVATE is assumed.

M=m

File access mode permitted to other users if the file is semiprivate or public, and if explicit access permission has not been granted to that user. If M=m is omitted, M=WRITE is assumed.

If m is:	Other users can attach the file in the following modes (refer to the ATTACH statement description):
E or EXECUTE	Execute.
R or READ	Read or execute.
RA or READAP	Read append, read, or execute.
RM or READMD	Read modify, read append, read, or execute.
A or APPEND	Append, read modify, read append, read, or execute.
M or MODIFY	Modify, append, read modify, read append, read, or execute.
W or WRITE	Write, modify, append, read modify, read append, read, or execute.
N or NULL	None.

Special care should be taken when using the read modify or read append mode. Programs using access techniques that do not expect concurrent updating of a file may get erroneous results if these modes are used.

CRM AAM (refer to the CYBER Record Manager Advanced Access Methods Reference Manual) does not anticipate concurrent updating of a file by another user. Therefore, if an attempt is made to alter and read a file by a concurrent user, a warning diagnostic message is issued stating that the file is bad when, in fact, it is not.

After a file is defined, it is always assigned to the job in write mode.

BR=br

Backup copy requirement; entries are tape (Y), MSF (MD), and no backup (N). If BR=br is omitted, a backup copy of the file is stored on tape.

PR=pr

Preferred file residence; entries are MSF (M) or no preference (N). If PR=pr is omitted, no preference is assumed.

PN=packnam

Name of an auxiliary device on which the direct access file is to reside. If PN=packnam is omitted, the file residence is determined by the PR, R, and S parameters.

R=r

Device type 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. If lfn_i already exists on a device other than that specified or if an illegal device is specified, a dayfile message so informs the user. If an auxiliary device name is not specified by the PN=packnam parameter or a previous PACKNAM control statement, the file is defined on a family device.

S=space — Decimal number of PRUs requested for the file. It cannot be larger than the user's validation limit (refer to LIMITS Statement in section 6). If no device has the specified amount of space available, a dayfile message so informs the user.

This parameter ensures that the file is assigned to a device that has the requested space available at the time the file is defined. It does not guarantee that the space will be available when the file is written.

NA — No abort option. If NA is specified, processing errors do not terminate the job.

If lfn_i does not exist, the device on which pfn_i resides depends on the R=r and S=space parameters.

R=r	S=space	File Residence
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, the file resides on that auxiliary device rather than a family device.

After the DEFINE statement has been processed, the new direct access file remains attached to the job in write mode. After the file is returned, the user must issue an ATTACH statement to access the direct access file. If the user purges an attached direct access file, the file remains attached to the job, although it has been removed from the user's permanent file catalog. Until the user returns the purged file, he cannot define a direct access file having the same local file name as the purged file.

GET STATEMENT

The GET control statement retrieves copies of indirect access permanent files for use as local files.

The control statement format is:

GET(lfn_1=pfn_1,lfn_2=pfn_2,...,lfn_n=pfn_n/UN=usernum,PW=passwrd,PN=packnam,R=r,NA)

lfn_i=pfn_i — Local file name lfn_i is the name given the retrieved copy of indirect access permanent file pfn_i. If lfn_i= is omitted, the local copy of the permanent file is called pfn_i. If no files are named, NOS uses the primary file name; the retrieved file copy is then the new primary file.

The full descriptions of the following optional parameters are given at the beginning of this section.

UN=usernum	User number. Specified if the permanent file(s) is in another user's catalog. The user must have permission to read or execute the file(s) (refer to SAVE Statement in this section). If only execute permission has been granted, the file is retrieved in execute-only mode.
PW=passwrd	File password. Specified if UN=usernum is specified, and if the permanent file has a password.
PN=packnam	Auxiliary device name. Specified if the permanent file(s) resides on an auxiliary device.
R=r	Device type. Specified if PN=packnam is specified, or if a PACKNAM control statement has been processed and the device type is other than the system default.
NA	No abort option. If NA is specified, processing errors do not terminate the job.

Each permanent file named must be an indirect access file. If the file, lfn_i, is assigned to the job before this statement is processed, it is returned. The new local file is rewound after it is retrieved. More than one user can have local copies of an indirect access file assigned to their jobs simultaneously.

If the user's current primary file is specified as an lfn on the statement, a copy of the associated permanent file, pfn, becomes the primary file. The time-sharing subsystem associated with the permanent file, pfn, becomes the job's current time-sharing subsystem (refer to the Network Products IAF Reference Manual or the NOS Time-Sharing User's Reference Manual).

If an auxiliary device has been previously specified by a PACKNAM statement, the system attempts to find pfn_i on the auxiliary device rather than on the family device.

OLD STATEMENT

The OLD control statement retrieves a copy of an indirect access permanent file and makes it the primary file.

The control statement format is:

OLD(lfn=pfn/UN=usernum,PW=passwrd,PN=packnam,R=r,NA,ND)

lfn=pfn	One- to seven-character file name lfn is given to the primary file copy of indirect access permanent file pfn. If lfn= is omitted, the primary file is named pfn.

The full descriptions of the following optional parameters (except ND) are given at the beginning of this section.

UN=usernum	User number. Specified if the indirect access permanent file is in another user's catalog.
PW=passwrd	File password. Specified if UN=usernum is specified, and if the permanent file has a password.

PN=packnam — Auxiliary device name. Specified if the permanent file resides on an auxiliary device.

R=r — Device type. Specified if an auxiliary device on a device type other than the installation-defined default is to be used.

NA — No abort option. If NA is specified, processing errors do not terminate the job.

ND — No drop option. If ND is specified, OLD changes the former primary file into a local file, but does not return any files. If ND is omitted, OLD returns all files assigned to the job.

If an auxiliary device has been previously specified by a PACKNAM statement, the system attempts to find the permanent file, pfn, on the auxiliary device rather than on the family device.

An OLD statement without the ND parameter releases all files assigned to the job. A copy of the indirect access permanent file named on the OLD statement becomes the primary file. The primary file is positioned at its BOI.

The primary file is rewound before every job step. Therefore, the file positioning statements, BKSP, SKIPEI, SKIPF, SKIPFB, and SKIPR, have no effect on the primary file. Also, when two copy statements are issued to write on the primary file, the second copy writes over the data written by the first copy because the primary file is rewound between copy statements.

PACKNAM STATEMENT

The PACKNAM control statement directs subsequent permanent file requests to the specified auxiliary device.

The control statement format is:

```
PACKNAM(PN=packnam)

    or

PACKNAM(packnam)
```

packnam — One- to seven-character name that identifies the auxiliary device to be accessed in subsequent permanent file requests.

PACKNAM allows the user to omit the PN keyword from control statement requests for files that reside on the specified packnam device. However, if permanent files on another auxiliary device are to be requested by a control statement, the PN keyword must be specified in the file request, or another PACKNAM request can be issued before the control statement. Refer to Mass Storage File Residence in section 2 for information concerning auxiliary permanent file devices.

The user cannot access permanent files residing on the family system devices while the PACKNAM request is in effect. To access these files, he must include a PACKNAM statement in either of the following formats.

```
PACKNAM.

    or

PACKNAM(PN=0)
```

PERMIT STATEMENT

The PERMIT control statement allows a user to explicitly permit another user to access a private file in his permanent file catalog. The PERMIT statement can also change the mode in which another user can access a semiprivate file.

The control statement format is:

PERMIT(pfn,usernum$_1$=m$_1$,usernum$_2$=m$_2$,...,usernum$_n$=m$_n$/PN=packnam,R=r,NA)

pfn	Name of the private or semiprivate file for which access permission is granted.
usernum$_i$=m$_i$	Specifies that user number usernum$_i$ is granted the access permissions indicated by access mode m$_i$. If m$_i$ is omitted, the read access mode is assumed. If m$_i$ is NULL, the user is explicitly denied permission to access the file. For the available access modes, refer to DEFINE Statement or SAVE Statement in this section.

The full descriptions of the following optional parameters are given at the beginning of this section.

PN=packnam	Auxiliary device name. Specified if the permanent file resides on an auxiliary device.
R=r	Device type. Specified if an auxiliary device on a device type other than the installation-defined default is to be used.
NA	No abort option. If NA is specified, processing errors do not terminate the job.

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(TY=ty,CT=ct,AD=ad,MD=md,CD=cd,AF,TM=tm,DN=dn,PN=packnam,R=r,NA)

TY=ty — File type to be purged.

ty	Action
I or INDIR	Purges all indirect access files.
D or DIRECT	Purges all direct access files.
A or ALL	Purges all files.

If this parameter is omitted, but other parameters are specified, the system assumes ty is ALL. To purge all files if no other parameters are specified, the user must specify TY=A.

CT=ct	File category to be purged; entries are PRIVATE, SPRIV, and PUBLIC.
AD=ad	Last access date; its format is yymmdd. All files last accessed before this date are purged, unless the AF parameter is specified.
MD=md	Last modification date; its format is yymmdd. All files last modified before this date are purged, unless the AF parameter is specified.
CD=cd	Creation date; its format is yymmdd. All files created before this date are purged, unless the AF parameter is specified.
AF	All files accessed after the date specified by the AD=ad parameter, modified after the date specified by the MD=md parameter, or created after the date specified by the CD=cd parameter are purged.
TM=tm	Time of day on the date specified by the AD, MD, or CD parameter; its format is hhmmss.
DN=dn	Device number assigned to a device during system initialization. Only files on that device are purged.
PN=packnam	Name of the auxiliary device on which the files to be purged reside. The PN parameter cannot be specified if a device number is specified (DN=dn).
R=r	Type of auxiliary device on which the files to be purged reside. The R parameter cannot be specified if a device number is specified (DN=dn).
NA	No abort option. If the specified auxiliary device is not available, the job is suspended until it becomes available.

To purge all files in his catalog, the user must enter

```
PURGALL(TY=A)
```

AF, CT, DN, NA, R, TY, TM, and one date (either AD, MD, or CD) can be entered on a single PURGALL statement.

PURGE STATEMENT

The PURGE control statement names files to be removed from the permanent file device.

The control statement format is:

PURGE(pfn_1,pfn_2,...,pfn_n/UN=usernum,PW=passwrd,PN=packnam,R=r,NA)

pfn_i	Name of a permanent file to be purged. If no file is named, and if a permanent file exists that has the same name as the primary file, that permanent file is purged; the primary file remains assigned to the job.

The full descriptions of the following optional parameters are given at the beginning of this section.

UN=usernum	User number. Specified if the file(s) to be purged is in another user's catalog. To purge a file, the user must have write permission for that file.
PW=passwrd	File password. Specified if UN=usernum is specified, and if the permanent file to be purged has a password.
PN=packnam	Auxiliary device name. Specified if the permanent file resides on an auxiliary device.
R=r	Device type. Specified if an auxiliary device on a device type other than the installation-defined default is to be used.
NA	No abort option. If NA is specified, processing errors do not terminate the job.

When a PURGE statement is issued for any direct access file, the file is purged and the permanent file catalog is altered accordingly. However, if the direct access file is attached to a job, it remains attached to the job until the user returns it.

If pfn_i does not exist, the following message is issued.

pfn NOT FOUND, AT nnn.

REPLACE STATEMENT

The REPLACE control statement can purge an indirect access permanent file and replace it with a copy of a local file on mass storage. It can also save a copy of a local file on mass storage as a new indirect access permanent file.

The control statement format is:

REPLACE(lfn_1=pfn_1,lfn_2=pfn_2,...,lfn_n=pfn_n/UN=usernum,PW=passwrd,PN=packnam,R=r,NA)

lfn_i=pfn_i	Specifies that a copy of local file lfn_i becomes an indirect access permanent file named pfn_i (one- to seven-character name). If an indirect access file named pfn_i already exists, it is replaced. If lfn_i is omitted, the name of the local file is assumed to be pfn_i. If no files are named, a copy of the primary file becomes an indirect access permanent file, replacing any existing indirect access permanent file by that name.

The full descriptions of the following optional parameters are given at the beginning of this section.

UN=usernum	User number. Specified if the file to be replaced is in another user's catalog. To replace another user's file, the user must have write permission and be validated to create indirect access permanent files (refer to LIMITS Statement in section 6).
PW=passwrd	File password. Specified if the UN=usernum is specified, and if the permanent file to be replaced has a password.

PN=packnam — Auxiliary device name. Specified if the permanent file to be replaced resides on an auxiliary device.

R=r — Device type. Specified if an auxiliary device on a device type other than the installation-defined default is to be used.

NA — No abort option. If NA is specified, processing errors do not terminate the job.

The local files, lfn_i, are rewound before and after the replace operation.

The indirect access file created has the same access category as the file it replaces. Permission information and alternate user access data for the file are retained when a file is replaced. If the file created is a new file, it is created as a private file.

SAVE STATEMENT

The SAVE control statement allows the user to retain a copy of a local file on mass storage as an indirect access file.

The control statement format is:

SAVE(lfn_1=pfn_1,lfn_2=pfn_2,...,lfn_n=pfn_n/PW=passwrd,CT=ct,M=m,SS=subsystem,
BR=br,PN=packnam,R=r,NA)

lfn_i=pfn_i — Specifies that a copy of local file lfn_i becomes an indirect access permanent file named pfn_i (one- to seven-character name). If lfn_i= is omitted, the name of the local file is assumed to be pfn_i. If no files are named, a copy of the primary file becomes an indirect access permanent file with the same name as the primary file.

The full descriptions of the following optional parameters (except M=m) are given at the beginning of this section.

PW=passwrd — One- to seven-character password that other users must specify to access the file.

CT=ct — File access category; entries are PRIVATE, SPRIV, and PUBLIC. If CT=ct is omitted, CT=PRIVATE is assumed.

M=m — File access mode permitted to other users if the file is public or semiprivate, and if explicit access permission has not been granted to that user. If M=m is omitted, M=WRITE is assumed.

If m is:	Other users can:
R or READ RA or READAP RM or READMD	Retrieve a copy of the file. This copy can be read or executed.
E or EXECUTE	Retrieve a copy of the file. This copy can only be executed.
A or APPEND	Append data to the file with the APPEND statement.

If m is:	Other users can:
M or MODIFY	Retrieve a copy of the file or append data to the file. The user can enter GET, OLD, NEW, and APPEND statements, but not a REPLACE statement, for the file.
W or WRITE	Retrieve a copy of the file, append data to it, replace it, or purge it.
N or NULL	No access is allowed.

SS=subsystem — Time-sharing subsystem associated with the file. If SS=subsystem is omitted, SS=NULL is assumed unless lfn is the primary file. In that case, the file is associated with the currently active subsystem. If SS is specified without a subsystem, the file is associated with the currently active subsystem.

BR=br — Backup copy requirement; entries are tape (Y), tape or MSF (MD), or no backup (N). If BR=br is omitted, a backup copy of the file is stored on tape. Because indirect access files are not stored on the MSF, BR=MD on a SAVE statement is equivalent to BR=Y.

PN=packnam — Name of the auxiliary device on which the indirect access file is to reside.

R=r — Device type on which the indirect access file is to reside. The device must be a permanent file mass storage device on which indirect access files are allowed.

NA — No abort option. If NA is specified, processing errors do not terminate the job.

The local files, lfn_i, are rewound before and after the save operation.

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 his job field length. The following statements are included in this category.

DMD	DMPECS	PBC
DMDECS	LBC	RBR
DMP	LOC	WBR

NOTE

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

The DMP and DMD control statements dump the exchange package or central memory in octal representation and/or display code equivalences. Likewise, the DMDECS and DMPECS control statements dump ECS memory. These statements are particularly helpful in creating dumps for debugging purposes (refer to section 12, Debugging Aids). 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.

DMD STATEMENT

The DMD control statement requests a dump similar to that of the DMP statement but adds the display code equivalences to the right of the octal representations. If lines are duplicated, they are suppressed and the following message is issued to the output file.

DUPLICATED LINES.

The control statement formats are:

```
DMD(fwa,lwa)
DMD(lwa)
DMD.
```

fwa — First word address of memory to be dumped; fwa is relative to RA. If fwa is absent, dump mode depends on the presence or absence of lwa.

lwa Last word address plus one of memory to be dumped; lwa is relative to RA. If lwa alone is present, DMD assumes fwa is 0. If neither fwa nor lwa is present, MDM dumps the exchange package and 40_8 locations before and after the program address in the exchange package. Only the lower 17 bits of the program address are used.

In batch jobs, the dump is written on file OUTPUT. Central memory dumps are written four words per line.

In time-sharing jobs, DMD is effective only within procedure files or when specified on the ENTER statement. A dump from a terminal is formatted for 72-column output and written on local file ZZZDUMP. DMD displays an informative message at the terminal. ZZZDUMP is not rewound before or after the dump.

DMDECS STATEMENT

The DMDECS control statement requests a dump of ECS memory on file OUTPUT. The dump is four words per line with display code equivalences to the right of the octal representations. If lines are duplicated, they are suppressed, and the following message is issued to the output file.

DUPLICATED LINES.

The control statement formats are:

DMDECS(fwa,lwa)
DMDECS(lwa)

fwa First word address of ECS memory to be dumped; fwa is relative to the reference address of the field in ECS being used by the job (RAE). If fwa is absent, DMDECS assumes fwa is 0.

lwa Last word address of ECS memory to be dumped; lwa is relative to RAE.

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

Dumping always stops at the field length in ECS (FLE) if lwa is greater than FLE. If either fwa or lwa is nonnumeric, the following error message is issued to the user's dayfile.

ARGUMENT ERROR.

If fwa is greater than FLE, fwa is set to FLE-10. If both fwa and lwa are greater than FLE, fwa is set to FLE-10 and lwa is set to FLE. If fwa is greater than lwa, the system issues the following message to the user's dayfile.

FWA .GE. LWA+1.

If neither fwa nor lwa is specified, the following message is issued to the user's dayfile.

ILLEGAL REQUEST.

If no ECS field length exists for the user, the following message is issued to the user's dayfile.

NO ECS.

The DMDECS statement can be used from a time-sharing terminal only in a procedure file and only after OUTPUT is assigned to mass storage, as in the following example.

```
.PROC,PROCB.
ASSIGN(MS,OUTPUT)
FTN(I=PROG)
LGO.
EXIT.
DMDECS(0,100)
ROUTE(OUTPUT)
```

DMP STATEMENT

The DMP control statement can request an exchange package dump or a central memory dump.

The control statement formats are:

```
DMP(fwa,lwa)
DMP(lwa)
DMP.
```

fwa — First word address of memory to be dumped; fwa is relative to the first word of the user's field length. If fwa is absent, the resulting dump depends on the presence or absence of lwa.

If fwa is greater than the user's field length, fwa is set at the user's field length minus 10_8. If fwa is greater than or equal to 400000_8, the first dump address is fwa minus 400000_8, memory from the first dump address through lwa is dumped, and the job is aborted.

lwa — Last word address plus one of memory to be dumped; lwa is relative to the first word of the user's field length. If lwa alone is present, DMP assumes fwa is 0. If neither fwa nor lwa is present, DMP dumps the exchange package and 40_8 locations before and after the address in the program address register in the exchange package. Only the lower 17 bits of the program address are used. If lwa is greater than the user's field length, the dump stops at the end of the field length.

If either fwa or lwa is nonnumeric, DMP dumps the exchange package and 40_8 locations before and after the program address register in the exchange package. If both fwa and lwa are greater than the user's field length, the last 10_8 words of the user's field length are dumped. If fwa equals lwa, the 10_8 words beginning at fwa are dumped. If fwa is greater than lwa, DMP issues an error message and terminates the job step.

The user must not place another control statement (other than DMP, DMD, DMPECS, DMDECS, or EXIT) between the program to be dumped and the DMP statement.

DMP suppresses duplicate lines and then issues the following output message.

```
DUPLICATED LINES.
```

In batch jobs, the dump is written on file OUTPUT. Central memory dumps are written four words per line.

In time-sharing jobs, DMP is effective only within procedure files or when specified on the ENTER statement. A dump from a terminal is formatted for 72-column output and written on local file ZZZDUMP. DMP displays an informative message at the terminal. ZZZDUMP is not rewound before or after the dump.

DMPECS STATEMENT

The DMPECS control statement dumps the contents of an ECS field length on file OUTPUT or a user-specified file. The dump is four words per line. If lines are duplicated, they are suppressed and the following notation is issued to the output file.

DUPLICATED LINES.

A DMPECS statement within a time-sharing job copies the contents of the ECS field length to the local file ZZZDUMP and displays a message at the terminal informing the user of the dump. ZZZDUMP is not rewound before or after the dump.

The control statement formats are:

DMPECS(fwa,lwa)
DMPECS(lwa)
DMPECS(fwa,lwa,f,lfn)

fwa	First word address of ECS memory to be dumped; fwa is relative to the reference address of the field in ECS being used by the job (RAE).
lwa	Last word of ECS memory to be dumped; lwa is relative to RAE.
f	Print format (included for compatibility with NOS/BE).
lfn	File to dump to.

If the first format is used, the field in ECS memory defined by fwa and lwa is dumped to the file OUTPUT. Display code equivalences do not appear.

If the second format is used, DMPECS assumes fwa is 0. Display code equivalences do not appear.

If the third format is used, the specified field in ECS is dumped to lfn. The parameter f is ignored. Display code equivalences appear to the right of the octal representations, the same as the DMDECS control statement.

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

Dumping always stops at the field length in ECS (FLE) if lwa is greater than FLE. If either fwa or lwa is nonnumeric, the following error message is issued to the user's dayfile.

ARGUMENT ERROR.

If fwa is greater than FLE, fwa is set to FLE-10. If both fwa and lwa are greater than FLE, fwa is set to FLE-10 and lwa is set to FLE. If fwa is greater than lwa, the system issues the following message to the user's dayfile.

FWA .GE. LWA+1.

If neither fwa nor lwa is specified, the following message is issued to the user's dayfile.

ILLEGAL REQUEST.

If no ECS field length exists for the user, the following message is issued to the user's dayfile.

NO ECS.

LBC STATEMENT

The LBC control statement is intended for loading binary data of unknown format.

The control statement format is:

LBC(addr)

addr — Address relative to RA at which binary load begins; if addr is omitted, 0 (RA) is assumed.

LBC reads only one record from file INPUT. The user must make an LBC call for each record of data to be loaded. If addr is specified in the program call, binary data is loaded beginning at that address; otherwise, loading begins at the reference address (RA).

LOC STATEMENT

The LOC control statement reads octal line images from file INPUT and enters them in the user's CM field length.

The control statement format is:

LOC(fwa,lwa)
LOC(lwa)
LOC.

fwa — First word address of an area to clear (zero) before loading correction statements. If fwa is absent, LOC assumes 0.

lwa — Last word address plus 1 of the area to be cleared. If lwa is absent, LOC assumes 0.

To process the LOC statement, the system reads correction statement images from the current INPUT record. A correction statement consists of an octal address and a data field. The address field specifies the location to be corrected, and the data field contains the data to be placed in that location. Both fields may start at any column as long as the address precedes the data. The address field consists of a one- to six-digit address. If it is 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 characters. If it is less than 20 characters, it is terminated by a nonblank, nonoctal character and is stored right-justified. If it is 20 characters, no terminator is required. Embedded blanks in the data field are ignored.

If both fwa and lwa are specified and both are nonzero, storage is cleared from fwa to lwa, and the octal line images are loaded at the specified addresses. If the current INPUT record is empty, LOC clears the indicated area of memory.

PBC STATEMENT

The PBC routine writes one record from the specified area of CM to file PUNCHB.

The control statement format is:

```
PBC(fwa,lwa)
PBC(lwa)
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 of the binary deck. If lwa alone is present, PBC assumes that fwa is RA. If lwa equals fwa, and a nonzero value is specified, PBC adds 10_8 to lwa. If fwa and lwa are 0 or are omitted, lwa is set to the value in the lower 18 bits of RA. If the upper 12 bits of RA are 7700_8, lwa is the lower 18 bits of the location following the prefix (77) table plus the length of the prefix table.

CM is not altered by PBC.

RBR STATEMENT

The RBR routine loads one binary record from a specified file.

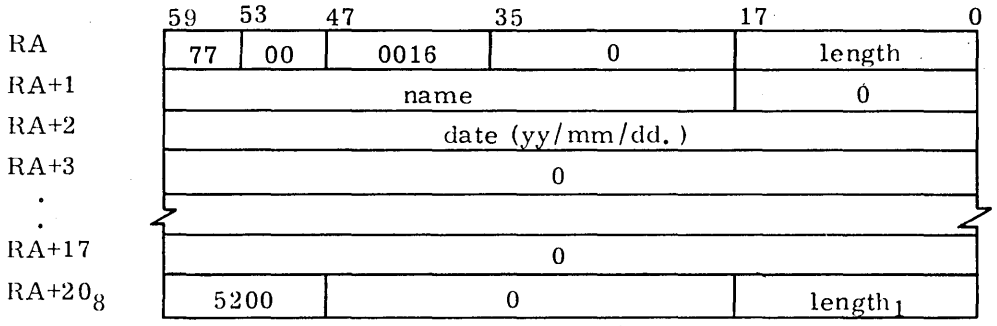
The control statement format is:

```
RBR(n,name)
```

n — n is used in constructing the name of the file containing the binary record to be read. If n is less than 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 — One- to seven-character name used in a record prefix.

The RBR routine loads one binary record from the specified file into central memory starting at RA. If the name parameter is included, a record prefix is placed in central memory starting at RA. The record itself follows. The following is the format of the record prefix.



length Record length including the prefix.
length$_1$ Record length minus words RA through RA+17$_8$.

If the record is too long for available memory, memory is filled, excess data is skipped, and the following message is issued to the user's dayfile.

RECORD TOO LONG.

WBR STATEMENT

The WBR routine writes a binary record from CM to a file at its current position.

The control statement format is:

WBR(n,rl)

n — n is used in constructing the name of the file on which the binary record is to be written. If n is less than 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.

TAPE MANAGEMENT 10

This section describes control statements used with magnetic tape files.† For additional information on NOS magnetic tape files, consult the glossary for definitions of terms; Magnetic Tape Files in section 2 for descriptions of tape labels and data formats; and appendix G for tape label formats. Section 6 describes the RESOURC statement required in jobs that use more than one tape or removable auxiliary pack concurrently.

NOTE

The term file as used in this section may refer to a multifile file. Refer to table 1-2-1 for the EOR and EOF marks for tape files.

The control statements described in this section are:

ASSIGN	Assigns a local file to a tape unit† (system origin jobs or jobs with system origin privileges only). Section 7 describes the ASSIGN statement for nontape files.
BLANK	Blank labels a tape and may restrict access to the labeled tape.
LABEL	Assigns a local file to a magnetic tape,† creates and verifies tape labels, and creates and accesses multifile set tapes.
LISTLB	Lists tape labels.
REQUEST	Assigns a local file to a magnetic tape device.†
VSN	Associates a file name with one or more VSNs for later assignment by a LABEL or REQUEST statement.

TAPE ASSIGNMENT

Whenever a tape is mounted, the system checks for labels. If the tape is labeled, the system records the volume serial number (VSN) read from the VOL1 label and the equipment on which the tape is mounted. When a tape assignment is requested by a LABEL or REQUEST statement specifying an lfn and a VSN (or an lfn that has been named in a previous VSN statement), the system compares the VSN with the VSNs read from mounted tapes. If a match is found, the system automatically assigns the tape to the requesting job, provided a deadlock would not occur.†† If the tape is not mounted, the system rolls out the job until a tape with the requested VSN is mounted.

† If the user does not specify a VSN parameter or an MT or NT parameter on the tape assignment statement, the operator can assign any device to the file.

†† Refer to the RESOURC statement in section 6.

For a mounted, unlabeled tape, the operator enters a command specifying the requested VSN. The system can then assign the tape. A VSN which contains nonalphanumeric characters should not be specified in a request for an unlabeled tape because nonalphanumeric characters cannot be entered with the operator command.

If a VSN is not associated with the requested lfn, the system directs the operator to assign an available device.

CONTROL STATEMENT RULES

On the tape assignment control statements (LABEL, REQUEST, and ASSIGN), the user can specify the tape label contents, tape density, track type, nine-track conversion mode, data format, noise size, and processing options. If any of these specifications are omitted, the system uses a default value.

NOTE

For nine-track tapes, the density specification given on the tape assignment is used only when the tape is written from load point. Otherwise, the tape is read or written using the density previously used for that tape. To ensure that a labeled tape is at load point for rewriting the tape at a new density, perform one of the following before the write operation.

- Rewind the tape.
- Specify the W parameter on the LABEL statement used to assign the tape.
- Assign the tape using a REQUEST or ASSIGN control statement.

Specification of duplicate or equivalent parameters is not allowed on tape assignment control statements.

NOTE

The user is advised not to create labeled S or L format tapes with tape marks embedded in the data. Future adherence to ANSI standards will make these tapes nonstandard as the ANSI standard allows tape marks to be used only as delimiters of label groups.

The system allows use of a continuation line for an ASSIGN, BLANK, LABEL, REQUEST, and VSN control statement when any one of these requires more than 80 characters. If, in processing one of these statements, the system does not encounter a termination character prior to the end of the line, it assumes the next line is a continuation line. A continuation line should be terminated with a valid terminator. The terminator for a continuation line must appear in or before column 80.

NOTE

The system accepts continuation lines from a time-sharing terminal only if they are within a procedure file.

The programmer can use literals for parameters that contain nonalphanumeric characters. These parameters are FI/L, FA, SI/M, VA, FA, OFA, and VSN. Nonalphanumeric characters are characters other than letters, numbers, and asterisks.

A literal is a character string delimited by dollar signs. Blanks within literals are retained. If the literal is to contain a dollar sign, two consecutive dollar signs must be included. Thus, the literal

```
$A B$$41$
```

is interpreted as

```
A B$41
```

When continuation lines are used, a literal cannot extend from one line to another.

PROCESSING OPTIONS

The PO= parameter on the LABEL, ASSIGN, and REQUEST tape assignment statements allows the user to specify one or more processing options that are to apply to that tape file. The characters representing the processing options and their meaning are listed below.

PO=S gives the default end-of-tape conditions. Default error recovery attempts to recover blocks having errors by repeatedly rereading the block. If the A, E, or N processing options are not specified, the program determines whether an error aborts the job or the program performs error processing (refer to the FET ep bit description in volume 2).

p_i	Description
A	Automatically aborts job on an irrecoverable read or write parity error (refer to the N option).
E	Error inhibit. All hardware read/write errors are ignored and processing continues. The system does not attempt error recovery, issue error messages, or return error status. During a read operation, blocks less than noise size (refer to the NS parameter) are unconditionally bypassed. This option is not intended for the normal user. It can be used to recover portions of data from a bad tape, to check out hardware, and to write on tape without skipping bad spots; in the latter case, the user is responsible for verifying that the data is written correctly.

Pi	Description
F	Force unload. Unload at end of usage. (Refer to the U option.)
G	Disables all hardware error correction activity in GE (6250 cpi) write mode. An on-the-fly error while writing a GE tape results in standard error recovery processing. The system erases the defective portion of tape, thereby reducing the amount of data that can be stored on the tape. The default is installation-defined (refer to the H option).
H	Enables hardware error correction activity in GE (6250 cpi) write mode. The system allows certain types of single track errors to be written that can be corrected when the tape is read (on-the-fly correction). This is the recommended mode because it provides efficient throughput, error recovery, and tape usage when writing GE tapes on a medium that is suitable for use at 3200 fci or 6250 cpi. The default option (G or H) is installation-defined.
I	Rewrites the block on which the end-of-tape occurred as the first block on the next volume, if the system senses the EOT during a write operation. During a read operation, the block on which the EOT occurred is ignored and reading continues on the next volume. If a tape mark and the EOT are sensed at the same time, the EOT is ignored. This option cannot be specified for I or SI format tapes. Refer to the P and S options.
L	Issues only the first and last error messages for each bad tape block. Numerous attempts are made to read each bad block, but only the messages for the first and last attempts are issued to the dayfile. The default is installation-defined (refer to the M option).
M	Issues an error message for each attempt to read a bad tape block. The default is installation-defined (refer to the L option).
N	Specifies that job is not automatically aborted on an irrecoverable read or write parity error (refer to the A option); data is passed to the job on a read operation without error status set even if the program requested error processing. This option is not intended for the normal user.
P	Writes a trailer sequence following the block on which the EOT was sensed, if the system senses the EOT during a write operation. Any data that occurs following the block on which EOT was sensed, yet before the tape mark, is ignored. During a read operation, the system transfers the block on which the EOT was sensed to the user job. The read operation resumes on the next reel. If a tape mark and the EOT are sensed at the same time, the EOT is ignored. Refer to the I and S options.
R	Enforce ring out. If the tape is mounted with the write ring in, job processing is suspended until the operator remounts the tape correctly.
S	Specifies where the system is to stop on an exit condition. For unlabeled tape, it directs the system to stop at the first tape mark after the EOT is sensed. For labeled tape, it directs the system to stop at the tape mark plus EOF1 or the tape mark plus EOV1 when the EOT is encountered. If, during a write operation, the system senses the end-of-tape, the system writes a trailer sequence following the block on which the EOT was sensed. This trailer sequence consists of a tape mark followed by an EOV1 label for labeled tapes and four tape marks for unlabeled tapes. The next block is written on the next volume. During a read operation, the EOT is noted and the system transfers to the user job the block on which the EOT was sensed plus all following blocks until a trailer sequence (as described previously) is recognized. Reading resumes on the next volume.

p_i	Description
U	Inhibit unload. Do not unload at the end of usage. For system origin jobs, the inhibit unload option is selected by default; for all other jobs, omission of the U option causes the tape to be unloaded at end of usage.
W	Enforce ring in. If the tape is mounted without the write ring in, job processing is suspended until the operator remounts the tape correctly.

If both ring enforcement options (R and W) are specified or more than one EOT option (I, P, or S) is specified, the system issues a dayfile message and terminates the job step.

For further information on end-of-tape/end-of-reel conditions, refer to the CLOSER, REWIND, and UNLOAD macros in section 3 and the LABEL macro in section 4 of volume 2.

ASSIGN STATEMENT

The ASSIGN control statement names a tape unit and the local file to be assigned to that unit. It can create an unlabeled tape file or access an existing labeled or unlabeled tape. It cannot create or verify tape labels.

NOTE

Only system origin jobs or users validated for system origin privileges (debug mode) and for use of magnetic tapes can use the ASSIGN statement to assign a tape unit.

Jobs that use this statement without proper validation are aborted, and a dayfile message is issued.

Before performing the assignment, the system unloads the local file (refer to the UNLOAD statement in section 7).

The following description applies only to magnetic tape files; for use of the ASSIGN statement with devices other than magnetic tape, refer to section 7.

The control statement format is:

ASSIGN(nn,lfn,VSN=$vsn_1/vsn_2=...=vsn_{n-1}/vsn_n$, {MT / NT}, {D=den / den}, F=format, LB=ℓ,
{FC=fcount / C=ccount}, CV=conv, NS=ns, PO=$p_1p_2...p_n$, {CK / CB})

Required parameters:

nn Device or device type to which the file lfn is assigned. nn can be the EST ordinal† of a magnetic tape unit or one of the device types MT or NT. Specifying MT informs the operator to assign the file to a seven-track magnetic tape drive; NT informs the operator to assign the file to a nine-track magnetic tape drive. Omission of this parameter results in an error.

† Contact installation personnel for a list of EST ordinals.

lfn Name of the file to be assigned to the device nn. Omission of this parameter results in an error.

Optional parameters:

VSN=$vsn_1/vsn_2=...=vsn_{n-1}/vsn_n$

One- to six-character volume serial number that uniquely identifies a reel of tape. ASSIGN does not use the VSN parameter to assign the tape. The nn parameter determines the tape assignment.

MT or NT

Specifies seven-track (MT) or nine-track (NT) tape drive. It must not conflict with the nn specification.

D=den or den

Tape density; must not conflict with the MT or NT specification. The default is installation-defined. The parameter is ignored for nine-track tapes not positioned at load point. Can be one of the following:

Seven-track (MT)		Nine-track (NT)	
den	Density	den	Density
LO	200 bpi	HD	800 cpi
HI	556 bpi	PE	1600 cpi
HY	800 bpi	GE	6250 cpi
200	200 bpi	800	800 cpi
556	556 bpi	1600	1600 cpi
800	800 bpi	6250	6250 cpi

F=format

Data format. Default is I. Refer to Magnetic Tape Files in section 2 for descriptions of the data formats.

- I Internal.
- SI System internal.†
- L Long block stranger.
- S Stranger.
- F Foreign.

LB=

Labeled or unlabeled tape. Default is KU if VSN is omitted or KL if VSN is specified.

- KU Unlabeled.
- KL ANSI-labeled. If the tape is a NOS tape, volume and header label access restrictions are enforced (refer to appendix G).
- NS Nonstandard-labeled. Assumes data begins immediately after the first tape mark.

† NOS/BE system default tape format (binary mode only); used for tape interchange with NOS/BE systems.

FC=fcount or C=ccount

Whenever F format is specified, this parameter must be specified. It specifies maximum block size in frames. No default value. Illegal for other tape formats.

CV=conv

Conversion mode† for nine-track tapes; applies to both labels and data on coded tapes; applies only to labels on binary tapes. Default is installation-defined. Parameter is ignored for unlabeled I or SI format binary tapes whose trailer labels are always ASCII. Must not be specified with MT or seven-track density specification.

- AS ASCII/display code conversion.
- US Same as AS.
- EB EBCDIC/display code conversion.

NS=ns

Noise size. Ignored for I and SI format tapes. Default is 18 frames for other formats. Maximum value is 31 frames. If NS=0 is specified, the default is used.

PO=$p_1p_2...p_n$

A string of characters (not separated by commas) that specify processing options (refer to Processing Options in this section).

CK or CB

lfn is to be used as a checkpoint file (refer to section 11).

- CK Each dump is written at the previous EOI of lfn.
- CB Each dump is written at the BOI of lfn.

Example:

```
ASSIGN(51,TAPE1,D=PE,F=SI)
```

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

BLANK STATEMENT

The BLANK control statement writes the ANSI standard labels VOL1, HDR1, and EOF1 following the load point of a tape. The labels are written as follows (asterisks represent tape marks):

	VOL1	HDR1	*	*	EOF1	*	*				

If the value of a labeled field is specified by a BLANK statement parameter, that value is written; otherwise, the default value is used. Refer to appendix G for the tape label formats and default values.

†Refer to Magnetic Tape Users in appendix A.

NOTE

A BLANK statement issued in a nonsystem origin job cannot overwrite a label containing an unexpired expiration date or a nonblank VA field.

If the FA field within the label is nonblank, a nonsystem origin job must specify the FA character using the OFA parameter. If the FA character is A, only the owner or a system origin job can overwrite the label.

The control statement format is:

BLANK(VSN=vsn, $\left\{\begin{matrix}MT\\NT\end{matrix}\right\}$, $\left\{\begin{matrix}D=den\\den\end{matrix}\right\}$,CV=conv,FA=fa,OFA=ofa,VA=va,

OWNER=usernum/familyname,LSL=lsl,U)

VSN=vsn

One- to six-character volume serial number that uniquely identifies the reel of tape. It is entered in the VOL1 label. It need not match the VSN previously recorded on the tape.

MT or NT

Specifies seven-track (MT) or nine-track (NT) tape drive. Installation-defined default. Must not conflict with D=den specification.

D=den or den

Tape density; must not conflict with the MT or NT specification. The default is installation-defined. den can be one of the following.

Seven-track (MT)		Nine-track (NT)	
den	Density	den	Density
LO	200 bpi	HD	800 cpi
HI	556 bpi	PE	1600 cpi
HY	800 bpi	GE	6250 cpi
200	200 bpi	800	800 cpi
556	556 bpi	1600	1600 cpi
800	800 bpi	6250	6250 cpi

CV=conv

Conversion mode† for nine-track tape labels. Installation-defined default. Must not be specified with MT or seven-track density specification.

AS ASCII/display code conversion.

US Same as AS.

EB EBCDIC/display code conversion.

†Refer to Magnetic Tape Users in appendix A.

FA=fa — File accessibility character indicating who has access to the labeled tape. Value entered in HDR1 and EOF1 labels.

- Blank — Unlimited access (default).
- A — Only the owner of this NOS written tape can access it.
- Other — In all future accesses of this tape, the user must specify this character.

OFA=ofa — Old file accessibility character on a labeled tape that is to be relabeled. This parameter must be specified if the FA field is currently other than A or blank. Future accesses of the tape must specify the character specified with the FA parameter.

VA=va — Volume accessibility character indicating that the volume must be accessed as an ANSI-labeled tape (LB=KL). If VA is nonblank, only a system origin job can destroy VOL1 (for example, assign tape as unlabled). Default is unrestricted access. Refer to the VOL1 format in appendix G.

OWNER=usernum/familyname — Owner identification entered in VOL1 label. Determines the owner for file accessibility (FA) parameter.

LSL=lsl — Label standard level entered in VOL1 label. Default is 1.

- 1 — Tape labels and data format for this volume conform to the ANSI standard.
- Blank — Tape labels and data format for this volume may or may not conform to the ANSI standard.

U — If U is specified, the tape is physically unloaded when returned after blank labeling. If U is omitted, physical unloading is inhibited. This parameter does not apply to system origin jobs.

An installation can use the BLANK statement to restrict use of its labeled tapes. Once a tape has been blank labeled, the user can modify the labels as follows:

1. If the volume accessibility field of VOL1 indicates unlimited access (that is, VA is blank), the user can:
   - Include another BLANK statement to change VOL1, HDR1, or EOF1 values.
   - Request the tape as unlabeled (with the parameter LB=KU) and write it in whatever format the user specifies.
   - Include a LABEL statement to change HDR1 by specifying one or more of the parameters associated with that label and specifying the W parameter.

2. If the volume accessibility field is nonblank, the user can:

   - Include a LABEL statement to change HDR1. However, in requesting a tape in which VA is nonblank, the user must specify an ANSI-labeled tape (with the parameter LB=KL), and therefore, cannot change or destroy the VOL1 label.
   - If validated, submit a system origin job to change VOL1.

LABEL STATEMENT

Like ASSIGN and REQUEST statements, the LABEL control statement associates a file name lfn with a magnetic tape, usually identified by its VSN. Unlike the ASSIGN and REQUEST statements, the LABEL statement can create and verify tape labels. It can also position a multifile set for access to any of its existing files or for appending a new file. The LABEL statement can create and access unlabeled as well as labeled tapes.

NOTE

A LABEL statement cannot overwrite a label with an unexpired expiration date (refer to appendix G).

To write the labels that begin a labeled tape (refer to Magnetic Tape Files in section 2), the user should specify a write label (W) parameter. The W parameter always rewinds the tape to load point and rewrites the first label group. The label contents remain the same when a LABEL statement with the W parameter names an lfn already assigned to a tape file.

If the tape was not previously part of a multifile set (the SI field in the first HDR1 label is blank), then specification of the SI and QN=9999 parameters rewrites the initial tape labels.

To position the tape after any HDR1 label other than the first HDR1 label (multifile set only), the SI parameter must be specified. When SI is specified, the R and W parameters are ignored unless QN=1 and the first file on the tape is to be written. The system determines where to position the tape by matching the SI, FI, and QN parameter values (if specified) to the corresponding values in the HDR1 label. (The HDR1 label format is given in appendix G.)

To write the EOF1 and HDR1 labels between two files in a multifile set (refer to figure 1-2-2), the user specifies the SI and QN=9999 parameters. The W parameter is ignored if specified when appending the file (QN=9999). To ensure that all files in a file set have the same set identifier, an appended file is given the same file set identifier as the previous file in the file set regardless of the SI=setid specification.

If neither the MT nor NT parameter is specified and no VSN is named, the operator can assign the file to any equipment. The user must be validated for the assigned equipment or the job is terminated.

The control statement format is:

```
LABEL(lfn,VSN=vsn1/vsn2=...=vsnn-1/vsnn, {MT|NT},DEN=den,F=format,LB=ℓ,
{FC=fcount|C=ccount},CV=conv,NS=ns,PO=p1p2...pn, {CK|CB}, {SI=setid|M=setid'}
{SN=secno|V=secno}, {QN=seqno|P=seqno}, {FI=fileid|L=fileid}, FA=fa,G=genno,E=gvn,
{CR=cdate|C=cdate}, {RT=yyddd|T=ddd}, {W|R})
```

Required parameter:

lfn — Name of the file that resides or is to reside on magnetic tape. If lfn is already assigned to a mass storage file, processing continues with the next control statement. To assign a previously assigned lfn, the user must return lfn before its reassignment. If lfn is already assigned to a tape and R parameter is specified, the contents of the tape labels are compared to the statement parameter specifications. If the label verification fails, the job aborts. Omission of lfn results in an error.

Optional parameters:

VSN=vsn1/vsn2=...=vsnn-1/vsnn — One- to six-character volume serial number that uniquely identifies a reel of tape. If VSN is omitted, the operator assigns an available unit to lfn. Multiple VSNs can be specified if separated by / or = characters. If the VSNs are separated by the = character, LABEL assigns lfn to the first available VSN in the list. If the VSNs are separated by the / character, lfn is a multivolume file set, and LABEL assigns the volumes in the sequence given. If VSN=, VSN=0, or VSN=SCRATCH is specified, a scratch tape is assigned. If a scratch tape is unavailable, the job is suspended until a tape is available.

MT or NT — Requests seven-track (MT) or nine-track (NT) tape drive. Installation-defined default. Must not conflict with D=den specification.

D=den — Tape density; must not conflict with MT or NT specification. The default is installation-defined. The parameter is ignored for nine-track tape not positioned at load point. Can be one of the following:

Seven-track (MT)		Nine-track (NT)	
den	Density	den	Density
LO	200 bpi	HD	800 cpi
HI	556 bpi	PE	1600 cpi
HY	800 bpi	GE	6250 cpi
200	200 bpi	800	800 cpi
556	556 bpi	1600	1600 cpi
800	800 bpi	6250	6250 cpi

F=format

Data format. Default is I. Refer to Magnetic Tape Files in section 2 for descriptions of the data formats.

- I Internal.
- SI System internal.†
- S Stranger.
- L Long block stranger.
- F Foreign.

LB=

Labeled or unlabeled tape. Default is KL.

- KL ANSI-labeled.
- KU Unlabeled.
- NS Nonstandard-labeled. Assumes data begins immediately after the first tape mark.

FC=fcount or C=ccount

Whenever F format is specified, this parameter must be specified. It specifies the maximum block size in frames (no default value). Illegal for other tape formats.

CV=conv

Conversion mode†† for nine-track tapes; applies to both labels and data on coded tapes; applies only to labels on binary tapes. Installation-defined default. Ignored for unlabeled I or SI format binary tapes whose trailer labels are always ASCII. Must not be specified with MT or seven-track density specification.

- AS ASCII/display code conversion.
- US Same as AS.
- EB EBCDIC/display code conversion.

NS=ns

Noise size; any block containing fewer than ns frames is considered noise and discarded. Ignored for I and SI format tapes. Default is 18 frames for other formats. Maximum value is 31 frames. If NS=0 is specified, the default is used.

PO=$p_1p_2 \cdots p_n$

A string of characters (not separated by commas) that specifies processing options. Refer to Processing Options in this section.

---

† NOS/BE system default tape format (binary mode only); used for tape interchange with NOS/BE systems.

†† Refer to Magnetic Tape Users in appendix A.

CK or CB — lfn is to be used as a checkpoint file (refer to section 11).

- CK — Each dump is written at the previous EOI of lfn.
- CB — Each dump is written at the BOI of lfn.

Optional tape label parameters (refer to appendix G):

SI=setid — One- to six-character file set identifier; must be specified for file positioning within a multifile set.

A file set identifier should be specified when the first file of a file set is written. When appending a file to a file set, the SI=setid parameter must be specified to position the multifile set, but the specified set identifier is not written in the HDR1 label. The appended file is given the same set identifier as the previous file in the file set.

If the SI-setid parameter is omitted when the first file of a file set is written, the set identifier field in the HDR1 label is left blank. A blank set identifier field is then written in the HDR1 labels of all files in the file set.

SN=secno — One- to four-digit file section number specifying the position of the volume within a multivolume file set (numbered consecutively from 0001). The default is 1.

QN=seqno — One- to four-digit file sequence number specifying the position of the file within the multifile set (numbered consecutively from 0001). The default is 1. QN must be set to 9999 to append a new file to a multifile set.

FI=fileid or L=fileid — A one- to seventeen-character file identifier recorded in the HDR1 label (refer to appendix G). The default is blank.

FA=fa — File accessibility character indicating who has access to the labeled tape.

- Blank — Unlimited access (default).
- A — Only the owner of the tape can access it.
- Other — To access the tape, the user must specify the character in the FA field of the HDR1 label.

G=genno — One- to four-digit generation number. The number zero cannot be used. The default is 1.

E=gvn — One- to two-digit generation version number. The default is 0.

CR=cdate — Creation date in the form yyddd where $1 \leq ddd \leq 366$. Used only on read operations; write operations always use the current date.

RT=yyddd or T=ddd

RT=yyddd specifies the expiration date where yy is the last two digits of the year, and ddd is the day of the year (1 ddd 366). T=ddd specifies the number of days the file is to be retained (0 ddd 999). The expiration date is entered in the HDR1 label. On or after this date the label and the file can be overwritten.

R or W

If R is specified, the system compares the values recorded on the file labels with the LABEL statement parameter values. If the comparison fails, it terminates the job. R is the default.

If W is specified, the system writes ANSI standard labels on the tape. The labels contain the values specified with the LABEL statement parameters or their default values. If the tape is mounted without the write ring, job processing is suspended until the operator remounts the tape correctly. If both the W and the PO=R parameters are specified, the job step aborts.

W and R are ignored when SI is specified and QN 1. When QN = 1 (default value) and W are specified, the initial header label is rewritten.

Example 1 - Reading and Writing a Single-File File Set:

In the following job, the user reads data from one tape and writes data on another tape. Program input data previously written at 1600 cpi on a nine-track tape unit is read from tape TP01. Output data is written on tape TP02.

```
FTNJOB.
USER(USRNAME,PASSWRD,FAMNAME)
CHARGE(CHRGNUM,PROJNUM)
RESOURC(PE=2)
FTN.
LABEL(TAPE1,VSN=TP01,D=PE,PO=R)
LABEL(TAPE2,VSN=TP02,D=PE,PO=W,W)
LGO.
/EOR
      PROGRAM SORT(INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT)
                     .
                     .
                     .
      END
/EOR
```

The RESOURC statement schedules two nine-track, 1600-cpi tape units for concurrent use in the job. The tapes are not requested until after compilation of the program in case compilation errors occur.

Assuming the compilation completes without fatal error, the LABEL statements request that two tapes, TP01 and TP02, be mounted and assigned to the job. The tape files are called TAPE1 and TAPE2 within this job. PO=R is specified for the input tape, ensuring that the tape does not have a write ring. PO=W is specified for the output tape which requires a write ring. The W parameter on the second LABEL statement specifies the writing of ANSI standard labels. Default values are used in the label fields.

Following completion of the job, the tapes are rewound and unloaded.

Example 2 - Reading and Writing a Multifile Set:

The following job writes the object programs produced by three compilations as three files of a multifile set. It then copies one of the files to mass storage and executes it.

```
SJOB.
USER(USRNAME,PASSWRD,FAMNAME)
CHARGE(CHRGNUM,PROJNUM)
GET,BSORT,QSORT,LSORT.
LABEL(STAPE,VSN=TP03,D=PE,SI=BINSET,FI=BSORT,PO=W,W)
FTN,I=BSORT,B=STAPE.
LABEL(STAPE,VSN=TP03,D=PE,SI=BINSET,FI=QSORT,QN=9999)
FTN,I=QSORT,B=STAPE.
LABEL(STAPE,VSN=TP03,D=PE,SI=BINSET,FI=LSORT,QN=9999)
FTN,I=LSORT,B=STAPE.
LABEL(STAPE,VSN=TP03,D=PE,SI=BINSET,QN=2)
COPYEI,STAPE,QSORTB.
QSORTB.
```

The GET statement retrieves three indirect access files - BSORT, QSORT, and LSORT - containing FORTRAN Extended source programs. The first LABEL statement requests the mounting and assignment of tape TP03 to the job. The W parameter specifies the writing of ANSI standard labels. The set identifier and file identifier fields are written using the values specified on the SI and FI parameters. The B=STAPE parameter on the FTN statement specifies that the object program is to be written on the tape file.

The QN=9999 parameter on the second and third LABEL statements specifies that a file is to be appended to the tape. Although the same set identifier is used for all files within a file set, the SI parameter must be specified when positioning a multifile set. The second and third compilations write the second and third files of the file set.

The QN=2 parameter on the fourth LABEL statement positions the tape at the second file of the file set. The tape could also have been positioned according to the file identifier (FI=QSORT). The second file is copied to a local mass storage file, QSORTB, and executed.

Example 3 - Replacing a File within a Multifile Set:

To replace a file within a multifile set, the user must first copy to temporary storage the files that follow the file to be replaced, then write the replacement file, and last rewrite the succeeding files in the file set.

```
RWJOB.
USER(USRNAME,PASWRD,FAMNAME)
CHARGE(CHRGNUM,PROJNUM)
GET,SORT2.
LABEL(STAPE,VSN=TP03,D=PE,SI=BINSET,QN=2,PO=W)
COPYEI(STAPE,SCRATCH)
LABEL(STAPE,VSN=TP03,D=PE,SI=BINSET,QN=1)
COPYEI(SORT2,STAPE)
LABEL(STAPE,VSN=TP03,D=PE,SI=BINSET,FI=QSORT,QN=9999)
COPYEI(SCRATCH,STAPE)
```

The first LABEL statement requests tape TP03 containing the multifile set created in example 2. The tape is mounted with a write ring inserted (PO=W) and is positioned at the second file of the file set (QN=2). The COPYEI statement copies the second file to the mass storage file, SCRATCH.

The second LABEL statement positions the tape at the first file. Although the contents of the first file are rewritten, the file labels remain unaltered.

The third LABEL statement appends the second file to the file set (QN=9999). The labels are rewritten so the user can change the label contents. The third COPYEI statement writes the second file as stored on file SCRATCH.

The third file of the multifile set is lost, because it was not saved before the tape was rewritten.

LISTLB STATEMENT

The LISTLB control statement lists the labels of an ANSI-labeled tape file previously assigned the file name lfn.

The control statement format is:

LISTLB(lfn, {SI=setid / M=setid}, {QN=seqno / P=seqno}, LO=ltype, L=out)

lfn — File name assigned to tape file whose labels are to be listed. Default is file name TAPE.

SI=setid or M=setid — One- to six-character file set identifier. If specified, only label groups whose HDR1 label contains this value are listed.

QN=seqno — One- to four-digit file sequence number. If seqno is specified, only the label group whose HDR1 label contains this value is listed. If seqno is specified, SI must be specified; otherwise, LISTLB terminates.

LO=ltype — Label type(s) to be listed. The default is R. Required and optional labels are listed in appendix G. Combinations of ltype mnemonics can be specified, such as LO=VH to list only the VOLn and HDRn labels.

- A List all labels.
- R List required labels.
- O List optional labels.
- V List VOLn labels.
- H List HDRn labels.
- F List EOFn labels.
- E List EOVn labels.
- U List UVLn, UHLn, and UTLn labels.

L=out — File on which the labels are to be listed. Default is OUTPUT.

To list labels for a multifile set (lfn contains more than one HDR1/EOF1 label pair), the tape must be positioned at load point. LISTLB then positions the tape for reading the requested labels. It searches for labels through all volumes associated with lfn. At the end of the multifile set or if an expected label group is not found, the following dayfile message is issued. n is the sequence number of the last file found. (nnn should be ignored.)

```
MULTI-FILE NOT FOUND, lfn AT nnn.
REQUEST SECTION n+1.
FOUND SECTION n.
```

After issuing this dayfile message, LISTLB leaves the tape positioned after the last listed label. The next statement processed for the tape file must be either RETURN, EVICT, UNLOAD, or LABEL.

Example 1:

The following statements list the second label group of file set ABCDEF.

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

Example 2:

To list only the volume and header labels (trailer labels omitted) of a multivolume file set, the user must request a volume of the file set, list its labels, and return the file set, repeating the procedure for each volume of the file set.

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

Example 3:

To list all labels of the following file set, only one LISTLB control statement is required.

```
VSN(T=REEL1/REEL2)
LABEL(T,VSN=REEL1,D=HY)
LISTLB(T)
```

The LISTLB(T) statement lists all labels on the volumes associated with T, that is, REEL1 and REEL2.

REQUEST STATEMENT†

The REQUEST control statement associates a file name, lfn, with a magnetic tape device,†† usually described in a comment following the statement terminator. This comment is displayed at the system console, directing the operator to make the requested assignment. However, if the tape is labeled and the user previously specified a VSN via the VSN control statement or included the VSN parameter on the REQUEST statement, the system can automatically assign the tape.

The REQUEST statement can create unlabeled tape files and access existing labeled and unlabeled tape files. It cannot create or verify tape labels.

The control statement format is:

REQUEST(lfn,VSN=$vsn_1/vsn_2=...=vsn_{n-1}/vsn_n$, {MT / NT}, {D=den / den}, F=format,
LB=ℓ, {FC=fcount / C=ccount}, CV=conv, NS=ns, PO=$p_1p_2...p_n$, {CK / CB}) comment

Required parameter:

lfn — Name of the file that resides or is to reside on magnetic tape. If lfn is already assigned to a mass storage file, processing continues with the next control statement. To assign a previously assigned lfn, the user must return lfn before its reassignment. Omission of this parameter results in an error.

Optional parameters:

VSN=$vsn_1/vsn_2=...=vsn_{n-1}/vsn_n$ — One- to six-character volume serial number uniquely identifying a reel of tape. The user should specify a VSN for labeled and unlabeled tapes. If VSN is omitted, the operator must assign an available device to lfn.

If VSN=, VSN=0, or VSN=SCRATCH is specified, a scratch tape is assigned. If a scratch tape is unavailable, the job is suspended until a tape is available.

Multiple VSNs can be specified if separated by a / or = character. If the VSNs are separated by the = character, the system assigns lfn to the first available VSN in the list. If the VSNs are separated by the / character, lfn is a multivolume file set, and LABEL assigns the volumes in the sequence given.

MT or NT — Requests seven-track (MT) or nine-track (NT) tape drive. Installation-defined default. Must not conflict with D=den specification.

---

†The user should employ the LABEL control statement for this operation.

††If the user does not specify a VSN parameter or an MT or NT parameter on the statement, the operator can assign any device to the file. If the user is not validated for the assigned device, the job aborts.

D=den or den

Tape density; must not conflict with MT or NT specification. The default is installation-defined. The parameter is ignored for nine-track tape not positioned at load point. Can be one of the following.

Seven-track (MT)		Nine-track (NT)	
den	Density	den	Density
LO	200 bpi	HD	800 cpi
HI	556 bpi	PE	1600 cpi
HY	800 bpi	GE	6250 cpi
200	200 bpi	800	800 cpi
556	556 bpi	1600	1600 cpi
800	800 bpi	6250	6250 cpi

F=format

Data format. Default is I. Refer to Magnetic Tape Files in section 2 for descriptions of the data formats.

I Internal.

SI System internal.†

S Stranger.

L Long block stranger.

F Foreign.

LB= ℓ

Labeled or unlabeled tape. Default is KL if a volume serial number is specified by the VSN parameter or by a VSN control statement; otherwise, the default is KU.

KL ANSI-labeled.

KU Unlabeled.

NS Nonstandard-labeled. Assumes that data begins immediately after the first tape mark.

FC=fcount or C=ccount

Whenever F format is specified, this parameter must be specified. It specifies the maximum block size in frames (no default value). Illegal for other tape formats.

CV=conv

Conversion mode †† for nine-track tapes; applies to both labels and data on coded tapes; applies only to labels on binary tapes. Installation-defined default. Ignored for unlabeled I or SI format binary tapes whose trailer labels are always ASCII. Must not be specified with MT or seven-track density specification.

AS ASCII/display code conversion.

---

†NOS/BE system default tape format (binary mode only); used for tape interchange with NOS/BE systems.

††Refer to Magnetic Tape Users in appendix A.

US Same as AS.

EB EBCDIC/display code conversion.

NS=ns — Noise size. Ignored for I and SI format tapes. Default is 18 frames for other formats. Maximum value is 31 frames. If NS=0 is specified, the default is used.

PO=$p_1p_2...p_n$ — A string of characters (not separated by commas) that specifies processing options. (Refer to Processing Options in this section.)

CK or CB — File lfn is to be used as a checkpoint file (refer to section 7 and section 11).

CK Each dump is written at the previous EOI of lfn.

CB Each dump is written at the BOI of lfn.

comment — The comment is displayed at the system console. In the comment field the user directs the operator to make the requested assignment.

Example:

To send a message to the operator requesting that volume XYZ be mounted on tape unit NT62 and assigned to lfn TAPE1, the user could issue the following statement.

```
REQUEST,TAPE1. NEED VSN=XYZ ON NT62.
```

VSN STATEMENT

The VSN control statement associates a file name lfn with one or more volumes of tape.† An lfn/VSN association allows the system to assign the specified VSN to lfn without reference to a VSN parameter on the LABEL or REQUEST statement or to an operator command. Once declared, an lfn/VSN association remains until the file is returned by an operation such as an EVICT, RETURN, or UNLOAD statement.

The control statement format is:

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

lfn_i — File name to be associated with vsn_i. This parameter is required if parameters are specified.

vsn_i — One or more one- to six-character volume serial numbers to be associated with lfn_i. If vsn_i contains nonalphanumeric characters, it must be a literal delimited by dollar signs ($).

vsn_i	Meaning
Omitted	An available scratch tape is automatically assigned to lfn_i.

†Up to 60 VSNs can be specified for a single file name in any combination of duplicate reel and/or multireel specifications.

vsn_i	Meaning
0	Same as omitted.
SCRATCH	Same as omitted.
$vsn_a=vsn_b=...=vsn_z$	Names duplicate volumes, any of which may be used with lfn_i.
$vsn_a/vsn_b/.../vsn_z$	Successive volumes to be assigned to lfn_i.† The system assigns volumes in the order listed.

With a VSN statement the user can:

- Omit the VSN keyword from his LABEL or REQUEST statements and specify lfn/VSN associations on the VSN statement instead. This allows the user to specify new VSNs without changing LABEL or REQUEST statements.

- Override the VSN specified on subsequent ASSIGN, LABEL, REQUEST, or VSN statements. For example, the sequence

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

  directs the system to assign FILEA to the tape with VSN 123. However, by returning file lfn, the user can specify another lfn/VSN association. Thus, the following sequence directs the system to assign FILEA to the tape with VSN 124.

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

- Associate the VSNs of two or more duplicate volumes with one file name. For example, the following statement indicates that either the tape with VSN VOL100 or the tape with VSN VOL101 can be assigned to FILE1.

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

- Specify the VSNs of a multivolume file set. For example, the following statement indicates that FILE2 may extend through the three volumes identified by VSN23, VSN24, and VSN25.

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

- Specify alternate volumes within a multivolume file set.

  ```
  VSN(FILE3=VSNA=VSN1/VSN2/VSNB=VSN3=VSN4)
  ```

  The first volume of the set can be either VSNA or VSN1. The second volume is VSN2. The third volume can be either VSNB, VSN3, or VSN4, depending on which is available.

---

† All subsequent volumes must have the same characteristics as the first volume in the sequence. (Characteristics include labels, track type, density, and conversion mode.) It is recommended that all volumes be blank labeled (refer to the BLANK statement) before use in a multivolume sequence.

CHECKPOINT/RESTART 11

A job may terminate as the result of system, operator, or programmer error. For some jobs, it becomes more advantageous to accept the overhead of checkpoint procedures than to run the risk of losing the entire job output. The checkpoint/restart feature is implemented through the CKP control statement and the RESTART control statement.

NOTE

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

CKP STATEMENT

The CKP control statement causes a checkpoint dump to be taken.

The control statement format is:

CKP(lfn_1,lfn_2,...,lfn_n)

lfn_i Specifies a file to be included in the checkpoint dump. If no files are specified, all files local to the job at the time the CKP statement is processed are checkpointed.

Each time a CKP statement is processed, the system takes a checkpoint dump. The dump is written on the tape or mass storage checkpoint file specified on a REQUEST, ASSIGN, or LABEL control statement with the CK or CB parameter. The dump consists of a copy of the user's central memory, the system information used for job control, and the names and contents of all assigned files explicitly or implicitly identified by the CKP statement. These files are:

- INPUT, OUTPUT, PUNCH, PUNCHB, P8, CCCCCCO, and LGO. These files are always included in the checkpoint dump. CCL ZZZZZxx working files are also included if present.
- Common files, library type files, working copies of indirect access files, and some direct access files. If one of these types of files is specified on the CKP statement, it is included in the checkpoint dump, and all other files of that type are excluded. If no files are specified, all files of these types assigned to the job are included in the dump.

Each checkpointed file is copied according to the last operation performed on it. If the last operation was a write, the file is copied from the BOI to its position at checkpoint time; only that portion is available at restart time. The file is positioned at the latter point.

If the last operation was a read and the EOI was not detected, the file is copied from its position at checkpoint time to the EOI; only that portion is available at restart time. The file is positioned at the former point. If the last operation was a read and the EOI was detected, no copy is performed.

The exception to this rule is the type of operation performed on execute-only direct access files. If a dump is specified for this type of file, its name and associated system information are copied but the contents of the file itself is not copied. Thus, if the user attempts to resume from such a dump, RESTART is unable to retrieve that file and aborts. The user can avoid this by selecting the NA and FC options of the RESTART statement and retrieving the file himself.

If the checkpoint file is to reside on mass storage, the user must include a SAVE or DEFINE control statement in the checkpoint job and a GET or ATTACH control statement in the restart job.

If the checkpoint file is to reside on magnetic tape, care should be taken to use a labeled or nonblank tape. An unlabeled blank tape (one which has never been used) cannot be specified as the checkpoint file since the checkpoint program attempts to read the tape to determine the number of the last checkpoint. The tape subsystem then aborts the job with a blank tape read message.

The system numbers checkpoints starting at 1 and increases by 1 to a limit of 4095. At this point, a second cycle of numbering begins, again starting at 1. An example showing how to restart from a specific checkpoint is given in the RESTART control statement section.

RESTART STATEMENT

The RESTART control statement directs the system to restart a previously terminated job from a specified checkpoint.

The control statement format is:

RESTART(lfn,nnnn,x_i)

lfn — Identifies the checkpoint file; the user must have write permission to lfn.

nnnn — Number of the checkpoint from which to restart; if nnnn is *, the last available checkpoint on lfn is used; if nnnn is omitted, the first checkpoint is used. The nnnn parameter can be obtained from the CHECKPOINT nnnn COMPLETE messages issued to the user's dayfile in response to CKP control statements.

x_i — Any of the following in any order:

RI — If this parameter is included, the control statement file on lfn is not restored. The control statement file of this restart job at its current position is used instead. If this parameter is not included, the entire control statement file of the checkpointed job is restored and set to its position at checkpoint time; any control statements following RESTART are not processed.

NA — If this parameter is included, RESTART does not abort if a required file is not available. Also, if NA is included and a read parity error occurs in an attempt to obtain a file from checkpoint nnnn, RESTART selects checkpoint nnnn-1 if it is available.

FC — Normally RESTART restores all files included in the specified checkpoint. However, if this option is selected, RESTART first checks if a file is already local to the restart job. If it is, RESTART does not replace it with the file on the checkpoint dump.

The user must assign lfn to his job before the RESTART statement is processed. He must include a REQUEST, ASSIGN, or LABEL control statement if lfn resides on magnetic tape or a GET or ATTACH control statement if lfn resides on mass storage.

Checkpoint dumps are numbered in ascending order from 1 to 4095. When nnnn equals 4095, the numbering sequence begins again at nnnn equal to 1. The value of nnnn depends on the structure of the checkpoint file, as defined by the CK and CB parameters of the REQUEST, ASSIGN, or LABEL control statements.

If CK was specified when the checkpoints occurred, each dump is appended to the checkpoint file, and therefore, all dumps up to the time the job aborted are available for restart. The user may specify a particular checkpoint dump in the following manner.

Assume a CK file of the name CHKFILE is being used and checkpoint number 4095 has been passed. The job is terminated at checkpoint number 10 in the second cycle of numbering. To restart the job from checkpoint 4 of the second numbering cycle, the following control statements can be used.

SKIPR(CHKFILE,8196)	There are two records for every checkpoint, and 4098 checkpoints must be skipped to reach checkpoint 4 of the second numbering cycle.
COPYBR(CHKFILE,AA,2)	The fourth checkpoint is copied to file AA. At this point, file CHKFILE is not positioned correctly for subsequent checkpoints. If the user intends to continue checkpointing on this file, a BKSP,CHKFILE. statement should be included.
RESTART(AA...)	The job is restarted from file AA using the fourth checkpoint.

If the CB parameter was specified on the ASSIGN, LABEL, or REQUEST statement naming the checkpoint file, each dump is written over the preceding dump, and therefore, only the last dump is available. If two REQUEST, ASSIGN, or LABEL statements with CB specified are submitted, successive dumps are alternated between two files; therefore, the last two dumps are available.†

If the CK parameter is specified for alternate files or if more than two checkpoint files are specified, the system issues a dayfile message and aborts the job.

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 aborts even if the NA option was selected.

DEBUGGING AIDS 12

Some program errors prevent compilation or assembly of the source program; other errors prevent execution of the object program. A programmer determines the cause of a compilation error using the compiler diagnostics, a source listing, and the compiler reference manual. The cause of an execution error is often more difficult to determine. If the programmer cannot determine the cause of the error from the execution error message, he can use the CDC CYBER Interactive Debug Utility or interpret memory dumps to locate the cause. CYBER Interactive Debug is described in its reference manual (listed in the preface). This section describes central memory dumps and their use as a debugging aid.

A programmer can dump the job exchange package and locations within the job field length using the DMP and DMD control statements described in section 9. (Dump restrictions are described under Security Control in section 3.) Most CPU mode errors result in an exchange package dump.

A programmer interprets a memory dump using the load map and the compiler-generated symbolic reference map. Dump interpretation may also require knowledge of display code equivalences (appendix A), machine codes, and internal integer and floating point number representations (refer to the COMPASS Reference Manual).

EXCHANGE PACKAGE DUMPS

The user can dump a job's exchange package using a DMP or DMD statement within the job (refer to section 9). Figures 1-12-1 and 1-12-2 show actual exchange package dumps. The format of the first dump is produced by all CYBER 170 models except model 176; all CYBER 70 models; and all 6000 Series computer systems. The second dump format is produced only by the CYBER 170 Model 176 computer system.

The following are the exchange package fields and their contents.

Label	Contents
P	Program address at which execution stopped.
RA	Reference address; starting address of central memory field length.
FL	Field length in central memory.
EM†	Exit mode. Each bit set indicates that if this hardware-detected error occurs, the program aborts. The bit positions are numbered with 0 as the rightmost bit (each digit shown represents 3 bits).

Bit Position	Error
11	CM data error.††
10	Central memory control (CMC) input error.††

†Does not apply to CYBER 170 Model 176.
††Applies to all CYBER 170 models except model 176.

```
EXCHANGE PACKAGE.

P       242  A0  51760  B0      0      (A0)  0000 0000 0000 0000 0000
RA   622400  A1      1  B1      1      (A1)  0516 0420 0000 0000 0000
FL    52000  A2    114  B2     30      (A2)  0400 0005 0300 0007 7775
EM     7007  A3    574  B3      6      (A3)  5555 5555 5555 5555 5555
RAE       0  A4    557  B4     22      (A4)  7777 7777 7777 7777 7776
FLE       0  A5    573  B5      1      (A5)  1717 0631 4631 4631 4632
MA     3600  A6      1  B6   7776      (A6)  0516 0420 0000 0000 0000
             A7    277  B7  14657      (A7)  3232 3232 3206 0300 0171

X0   7777 7777 7777 0000 0000
X1   0000 0000 0000 0000 0000
X2   0000 0000 0000 0000 0000
X3   0000 0000 0040 0000 0000
X4   2000 0000 0000 0000 0012
X5   1717 0631 4631 4631 4632
X6   0516 0420 0000 0000 0000
X7   0000 0000 0000 0000 0000

(RA)    0000 0000 0000 0000 0000
(RA+1)  0516 0420 0000 0000 0000
```

Figure 1-12-1. Exchange Package Dump

```
EXCHANGE PACKAGE.

P     10435  A0   2165  B0      0      (A0)  1725 2420 2524 0000 0131
RA   136100  A1      1  B1      1      (A1)  0516 0420 0000 0000 0000
FL    15000  A2   6251  B2 777755      (A2)  1717 0631 4631 4640 3615
PSD   70000  A3      2  B3   6032      (A3)  0000 0000 0000 0000 0000
RAE       0  A4   6207  B4  11437      (A4)  0400 0062 4600 0000 0000
FLE       0  A5   4324  B5  12711      (A5)  2000 0000 0000 0000 0065
MA     1200  A6      1  B6 776677      (A6)  0516 0420 0000 0000 0000
EEA    1200  A7  12557  B7     30      (A7)  6000 0000 0000 0001 5000

X0   0000 0000 0000 0000 0000
X1   0000 0000 0000 0000 0000
X2   1717 0631 4631 4640 3615
X3   2000 0000 0000 0000 0012
X4   2000 0000 0000 0000 0000
X5   0000 0000 0000 0000 0000
X6   0516 0420 0000 0000 0000
X7   2000 0000 0000 0000 0001

(RA)    0000 0000 0000 0000 0000
(RA+1)  0516 0420 0000 0000 0000
```

Figure 1-12-2. Exchange Package Dump for CYBER 170 Model 176

Label	Contents	
	Bit Position	Error
	9	ECS flag register operation parity error.†
	5-8	Not used.
	3-4	Hardware error exit status bits.††
	2	Indefinite operand.
	1	Operand out of range.
	0	Address out of range.

The EM field in figure 1-12-1 has bit positions 11, 10, 9, 2, 1, and 0 set.

PSD††† Program status designator (PSD) register. Each bit set indicates the setting of a mode flag or an error condition. The bit positions are numbered with 0 as the rightmost bit (each digit shown represents 3 bits).

Bit Position	Error
14	Indefinite mode.
13	Overflow mode.
12	Underflow mode.
11	LCME error.
10	CM error.
9	LCME block range error.
8	CM block range error.
7	LCME direct range error.
6	CM direct range error.
5	Program range error.
4	Not used.
3	Step condition.
2	Indefinite condition.
1	Overflow condition.
0	Underflow condition.

†Applies to all CYBER 170 models except model 176.
††Applies to CYBER 70 Model 74 only.
†††Applies to CYBER 170 Model 176 only.

The PSD field in figure 1-12-2 has bit positions 14, 13, and 12 set.

RAE	ECS reference address; starting address of ECS field length.
FLE	ECS field length.
MA	Monitor address (normal exit address for the CYBER 170 Model 176).†
EEA	Error exit address (CYBER 170 Model 176).†
Ai	Contents of the address registers.
(Ai)	Contents of the central memory word addressed by the named address register.
Bi	Contents of the increment registers.
Xi	Contents of the operand registers.
(RA)	Contents of the reference address word.
(RA+1)	Contents of the request word following the reference address word.

USING DUMPS

A NOS user receives an exchange package and partial CM dump when a hardware-detected error occurs. He can also obtain dumps of his job's exchange package and field length by including a DMP or DMD statement in his job. FORTRAN users can generate a CM dump within a program using the DMP subroutine (refer to the FORTRAN Extended 4 or FORTRAN 5 Reference Manual). COMPASS users can specify the REPRIEVE macro to control error processing and the SYSTEM macro to generate dumps (refer to sections 10 and 11 of volume 2).

When the system hardware detects one of the error conditions listed in the MODE statement description (section 6), NOS dumps the job exchange package and the contents of the 32 words preceding and the 32 words succeeding the address where the job step terminated.

The user can specify control statements after an EXIT statement; the control statements then are processed only if the job step terminates abnormally (refer to Error Control in section 3). Users frequently specify a dump statement to be executed if the job step aborts. The resulting dump is analyzed to determine the cause of the job step abort.

Example 1 - Finding the Source Program Location Where the Program Terminated:

This example uses a dump to find the cause of an execution error.

The user submits the following job to compile and execute a FORTRAN Extended program called FPROG.

```
FJOB.
USER(USRNAME,PASSWRD,FAMNAME)
CHARGE(CHRGNUM,PROJNUM)
GET,FPROG.
FTN,I=FPROG.
MAP(PART)
LDSET(PRESET=ZERO)
LGO.
```

†Normally this does not apply to the applications programmer.

The FTN statement compiles the source program in the retrieved file FPROG. The resulting program listing and symbolic reference map are shown in figure 1-12-3. The fields within the reference map are defined in the FORTRAN Extended 4 Reference Manual.

```
          PROGRAM FPROG      73/74   OPT=1                    FTN 4.8+505       79/08/29. 11.03.07      PAGE     1

      1               PROGRAM FPROG (INPUT,OUTPUT)
                      DIMENSION N(10)
                      DATA (N(I),I=1,10)/1,2,3,4,5,6,7,8,9,10/
                      NSUM=0
      5               DO 10 I=1,111111
                   10 NSUM=NSUM+N(I)
                      STOP
                      END


       SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
 4137  FPROG

VARIABLES     SN  TYPE           RELOCATION
 4153  I         INTEGER                          4154  N         INTEGER    ARRAY
 4152  NSUM      INTEGER

FILE NAMES        MODE
    0  INPUT                  2054  OUTPUT

STATEMENT LABELS
    0  10

LOOPS  LABEL    INDEX     FROM-TO    LENGTH    PROPERTIES
 4144  10       I           5 6         3B     INSTACK

STATISTICS
  PROGRAM LENGTH              160B      112
  BUFFER LENGTH              4006B     2054
         52000B CM USED
```

Figure 1-12-3. Example 1: Program Listing and Symbolic Reference Map

The MAP(PART) statement instructs the loader to generate a partial load map when it loads the program. The LDSET(PRESET=ZERO) statement tells the loader to set uninitialized memory words to zero during the next load. (Refer to the CYBER Loader Reference Manual for loader statement descriptions.)

The LGO statement loads and executes the object program that the compiler wrote on file LGO. The partial load map is shown in figure 1-12-4. The fields within the load map are defined in the CYBER Loader Reference Manual.

The job dayfile is shown in figure 1-12-5. Program execution terminates abnormally, resulting in the following error message.

CM OUT OF RANGE.

As explained in appendix B, this message indicates that the program references an address outside the job field length.

```
LOAD MAP - FPROG                                CYBER LOADER 1.5-505      79/08/29. 11.03.09.      PAGE     1


FWA OF THE LOAD            111
LWA+1 OF THE LOAD         7446

TRANSFER ADDRESS -- FPROG           4250

PROGRAM ENTRY POINTS --       FPROG          4250


PROGRAM AND BLOCK ASSIGNMENTS.

BLOCK       ADDRESS   LENGTH    FILE      DATE      PROCSSR VER LEVEL  HARDWARE   COMMENTS

FPROG           111     4166    LGO       79/08/29 FTN      4.8 505    666X I     PROGRAM   OPT=1
/STP.END/      4277        1
/FCL.C./       4300       26
/Q8.IO./       4326      144
Q2NTRY=        4472        1 SL-FORTRAN  79/07/09 COMPASS 3.6 505                 FCL INITIALIZATION ROUTINE.
/FCL=ENT/      4473       40
FCL=FDL        4533       40 SL-FORTRAN  79/07/09 COMPASS 3.6 505                 FCL CAPSULE LOADING
FEIFST=        4573        3 SL-FORTRAN  79/07/09 COMPASS 3.6 505                  CONVERTED DATA STORAGE
FORSYS=        4576      515 SL-FORTRAN  79/07/09 COMPASS 3.6 505                 FORTRAN OBJECT LIBRARY UTILITIES.
FORUTL=        5313       25 SL-FORTRAN  79/07/09 COMPASS 3.6 505                 FCL MISC. UTILITIES.
GETFIT=        5340       64 SL-FORTRAN  79/07/09 COMPASS 3.6 505                 LOCATE AN FIT GIVEN A FILE NAME.
SYSAID=        5424        1 SL-FORTRAN  79/07/09 COMPASS 3.6 505                 LINK BETWEEN SYS=AID AND INITIALIZATION CO

CPUCPM         5425        5 SL-SYSLIB   79/06/09 COMPASS 3.6 498                 79/05/10.          CONTROL POINT MANAGER

CPU.SYS        5432       40 SL-SYSLIB   79/07/09 COMPASS 3.6 505                 PROCESS SYSTEM REQUEST.
CMF.ALF        5472      162 SL-SYSLIB   79/07/09 COMPASS 3.6 505                 CMM V1.1 - ALLOCATE FIXED.
CMF.CSF        5654        6 SL-SYSLIB   79/07/09 COMPASS 3.6 505                 CMM V1.1 - CHANGE SPECS FIXED.
CMM.FFA        5662       14 SL-SYSLIB   79/07/09 COMPASS 3.6 505                 CMM V1.1 - FIXED FREE ALGORITHM.
CMF.FRF        5676       36 SL-SYSLIB   79/07/09 COMPASS 3.6 505                 CMM V1.1 - FREE FIXED.
CMF.GSS        5734       22 SL-SYSLIB   79/07/09 COMPASS 3.6 505                 CMM V1.1 - GET SUMMARY STATISTICS.
CMM.KIL        5756       12 SL-SYSLIB   79/07/09 COMPASS 3.6 505                 CMM V1.1 - DEACTIVATE CMM.
CMM.MEM        5770        7 SL-SYSLIB   79/07/09 COMPASS 3.6 505
CMM.R          5777      206 SL-SYSLIB   79/07/09 COMPASS 3.6 505                 CMM V1.1 - RESIDENT SUBROUTINES.
CMF.SLF        6205       22 SL-SYSLIB   79/07/09 COMPASS 3.6 505                 CMM V1.1 - SHRINK AT LWA FIXED.
/FDL.COM/      6227       14
FDL.RES        6243      211 SL-SYSLIB   79/07/09 COMPASS 3.6 505                 FAST DYNAMIC LOADER RESIDENT.
FDL.MMI        6454      222 SL-SYSLIB   79/07/09 COMPASS 3.6 505                 FDL MEMORY MANAGER INTERFACE.
CTL$RM         6676      427 SL-SYSLIB   79/07/09 COMPASS 3.6 505                 CRM CONTROLLING ROUTINE.
ERR$RM         7325       25 SL-SYSLIB   79/07/09 COMPASS 3.6 505                 CRM ERROR PROCESSOR ENTRY.
LIST$RM        7352       67 SL-SYSLIB   79/07/09 COMPASS 3.6 505                 CRM - ALLOCATE SPACE FOR LIST OF FILES
RM$SYS=        7441        5 SL-SYSLIB   79/07/09 COMPASS 3.6 505                 CRM - POST RA+1 REQUEST


     .285 CP SECONDS              24000B CM STORAGE USED                    2 TABLE MOVES
```

Figure 1-12-4. Example 1: Partial Load Map

```
11.03.06.FJOB.
11.03.06.USER(USRNAME,,FAMNAME)
11.03.06.CHARGE(CHRGNUM,PROJNUM)
11.03.07.GET,FPROG.
11.03.07.FTN,I=FPROG.
11.03.08.       .093 CP SECONDS COMPILATION TIME
11.03.08.MAP(PART)
11.03.08.LDSET(PRESET=ZERO)
11.03.08.LGO.
11.03.09. CPU ERROR EXIT AT  004256.
11.03.09. CM OUT OF RANGE.
```

Figure 1-12-5. Example 1: Dayfile from a Job Run

The exchange package dump that results from the job step abort is shown in figure 1-12-6. The address in register A4 is 7500, the same value as that given for the job field length (FL). An A register containing an address greater than or equal to the job field length indicates that the job step attempts to reference data outside its field length. A bad address in registers A1 through A5 indicates the error occurs on a read request. A bad address in register A6 or A7 indicates an attempt to write outside the job field length.

The address where the program terminated and the type of error that occurred are given in the dayfile.

```
CPU ERROR EXIT AT 004256.
CM OUT OF RANGE.
```

This address can also be found in bits 48 through 30 in the RA field of the exchange package. Usually, the program actually terminates at the instruction word preceding the address given, in this case, word 4255. Bits 59 through 49 of the RA contain the value 1, indicating that a mode 1 error occurred (CM OUT OF RANGE).

To find the approximate object program address to which memory address 4256_8 corresponds, the user refers to the load map (figure 1-12-4). The load map lists each of the blocks loaded, its length, and where it was loaded. If overlays or capsules are used in the program, an area within the field length may contain different program blocks at different times during program execution. In that case, the user must also determine which overlay or capsule was in memory at program termination.

Program FPROG has only one program block, FPROG. The remainder of the load is from various system libraries. It is 4166_8 words long and was loaded at address 111_8 according to the load map. The relative address where the program terminated is therefore 4255_8-111_8, or 4144_8.

Referring to the symbolic reference map (figure 1-12-3), the user finds that relative address 4144_8 is the address of loop label 10. Loop label 10 is on line 6 of the source program. By examining that loop, the user can determine that the execution error occurred when loop index I was set too large.

The symbolic reference map generated by the loader does not provide relative addresses for each line of source code. In the usual case, the user is not given the exact source code location corresponding to the termination address. The user must approximate the termination location as being between two relative addresses given in the reference map.

Example 2 - Finding the Contents of a Program Variable:

The user can also use a CM dump to determine the contents of a program variable when program execution terminated. For example, assume that the user submitted FJOB after correcting the source code and adding the statement, DMP,5000. The resulting symbolic reference map and load map are identical to those shown in figures 1-12-3 and 1-12-4. To determine the contents of the variable NSUM following normal program termination, the user must determine its absolute address.

The relative address of NSUM as given in the reference map in figure 1-12-3 is 4152_8. The block containing the object program, FPROG, was loaded at absolute address 111_8 so the absolute address of NSUM is 111_8+4152_8, or 4263_8.

The user then finds address 4263_8 in the memory dump. (Portions of the dump are shown in figure 1-12-7.) The content of address 4263_8 is

```
00000 00000 00000 00067
```

Because NSUM is an integer variable, the number is stored in integer format. Converted to base 10, this value is 55 (the sum of the numbers 1 through 10).

```
   EXCHANGE PACKAGE.

   P         0  A0       0  B0       0      (A0)  0001 0042 5600 0000 0000
   RA   375500  A1       0  B1       0      (A1)  0001 0042 5600 0000 0000
   FL     7500  A2       0  B2       0      (A2)  0001 0042 5600 0000 0000
   EM     7007  A3       0  B3       0      (A3)  0001 0042 5600 0000 0000
   RAE       0  A4    7500  B4       0      (A4)  0000 0000 0000 0000 0000
   FLE       0  A5    4263  B5       0      (A5)  0631 6514 3474 6057 1525
   MA     2600  A6    4264  B6  331007      (A6)  0000 0000 0000 0000 0001
                A7    4263  B7    3214      (A7)  0631 6514 3474 6057 1525

   X0   0000 0000 0000 0000 0000
   X1   0000 0000 0000 0000 0000
   X2   0000 0000 0000 0000 0000
   X3   0000 0000 0000 0000 0000
   X4   0000 0000 0000 0000 0000
   X5   0631 6514 3474 6057 1525
   X6   0000 0000 0000 0000 0001
   X7   0631 6514 3474 6057 1525

(RA)    0001 0042 5600 0000 0000
(RA+1)  0000 0000 0000 0000 0000


CM DUMP FROM    4216 TO    4316.

  4214      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000
        DUPLICATED LINES.
  4240      00000 00000 00000 00000      20020 00000 00000 04244      00000 00000 00000 11610      00000 00000 00000 00000
  4244      11162 02524 00000 00111      17252 42025 24000 02165      00000 00000 00000 00000      06202 21707 55550 04250
  4250      04000 04251 00000 00001      43700 71600 00001 46000      51700 04263 51600 04264      51500 04264 61603 31007
  4254      63750 46000 46000 46000      51500 04263 51470 04264      61770 00001 36745 54750      06670 04255 76770 46000
  4260      51700 04264 51100 04262      04000 04700 46000 46000      00000 00000 00000 00000      06316 51434 74605 71525
  4264      00000 00000 00000 00001      00000 00000 00000 00001      00000 00000 00000 00002      00000 00000 00000 00003
  4270      00000 00000 00000 00004      00000 00000 00000 00005      00000 00000 00000 00006      00000 00000 00000 00007
  4274      00000 00000 00000 00010      00000 00000 00000 00011      00000 00000 00000 00012      04000 04727 61000 46000
  4300      55555 55555 55555 55555      40404 04040 40404 04040      11162 02524 00000 00000      17252 42025 24000 00000
  4304      17171 27432 14774 13155      20001 20726 42717 30565      00000 00000 00000 00000      00000 00000 00000 04244
  4310      00000 00000 00000 11610      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000
  4314      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00024      00000 00000 00000 00000
```

Figure 1-12-6. Example 1: Exchange Package Dump

```
CM DUMP FROM       0 TO    5000.

     0      00000 00000 00000 00000      05160 42000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000
     4      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000
        DUPLICATED LINES.
    54      56110 03110 00054 54710      51100 00001 03110 00055      64550 02550 00000 46000      00000 00000 00000 00000
    60      15051 52000 00000 00061      00000 07500 00000 00001      00000 00000 00000 00000      00000 00000 00000 00000
    64      14071 70000 00000 00000      40000 00000 00000 07446      40000 00000 01000 00111      40000 00000 00000 00000
    70      14071 75700 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000
    74      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000
   100      54000 00000 01000 00001      00733 50000 00000 07446      00000 00000 00000 00000      00000 00000 00000 00000
   104      00000 00000 00000 07446      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000
   110      06202 21707 00000 04250      11162 02524 00000 00001      00000 00000 00000 00162      00000 00000 00000 00000
   114      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000
   120      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00600 00000
   124      00000 00022 00000 00000      00000 00000 00000 00000      14000 00000 00000 00000      00000 00000 00000 00000
   130      00000 00006 00000 00000      00000 00000 00000 00000      00000 00000 00000 02003      00000 00000 00000 00000
   134      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000
        DUPLICATED LINES.
  2164      00000 00000 00000 00000      17252 42025 24000 00001      00000 00000 00000 02236      00000 00000 00000 00000
  2170      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000
  2174      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00600 00000
  2200      00000 00022 00000 00000      00000 00000 00000 00000      14000 00000 00000 00000      00000 00000 00000 00000
  2204      00000 00006 00000 00000      00000 00000 00000 00000      00000 00000 00000 02003      00000 00000 00000 00000
  2210      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000      00000 00000 00000 00000
        DUPLICATED LINES.
  4240      00000 00000 00000 00000      20020 00000 00000 04244      00000 00000 00000 11610      00000 00000 00000 00000
  4244      11162 02524 00000 00111      17252 42025 24000 02165      00000 00000 00000 00000      06202 21707 55550 04250
  4250      04000 04251 00000 00001      43700 71600 00001 46000      51700 04263 51600 04264      51500 04264 61600 00012
  4254      63750 46000 46000 46000      51500 04263 51470 04264      61770 00001 36745 54750      06670 04255 76770 46000
  4260      51700 04264 51100 04262      04000 04700 46000 46000      00000 00000 00000 00000      00000 00000 00000 00067
  4264      00000 00000 00000 00013      00000 00000 00000 00001      00000 00000 00000 00002      00000 00000 00000 00003
  4270      00000 00000 00000 00004      00000 00000 00000 00005      00000 00000 00000 00006      00000 00000 00000 00007
  4274      00000 00000 00000 00010      00000 00000 00000 00011      00000 00000 00000 00012      04000 04727 61000 46000
  4300      55555 55555 55555 55555      40404 04040 40404 04040      11162 02524 00000 00000      17252 42025 24000 00000
```

Figure 1-12-7. Example 2: Central Memory Dump

NOS provides the following utilities for file maintenance.

EDIT	Edits a text file.
KRONREF	Generates a cross-reference listing of system symbols.
MODIFY	Edits a Modify-formatted program library file.
OPLEDIT	Removes modification decks and identifiers from a Modify-formatted program library file.
PROFILE	Enables a master user to update and inquire about a profile file.
UPDATE	Edits an Update-formatted program library file.
UPMOD	Converts an Update-formatted program library file to a Modify-formatted program library file.
XEDIT	Edits a text file.

EDIT STATEMENT

The EDIT control statement calls the Text Editor utility. The Text Editor enables a user to manipulate data on a specified mass storage file through use of special input directives called edit commands. For a detailed description of the Text Editor and an explanation of these commands, refer to the Text Editor Reference Manual.

The control statement format is:

EDIT(lfn_1,m,lfn_2,lfn_3)

or

EDIT(FN=lfn_1,M=m,I=lfn_2,L=lfn_3)

lfn_1	Name of file to be edited (referred to as edit file). This specification is required for batch origin jobs.
m	Mode of file processing: ASCII or AS — ASCII mode edit file. NORMAL or N — NORMAL mode edit file.
lfn_2	File from which directives (edit commands) are to be read. If omitted, INPUT is assumed.
lfn_3	File to which output is to be written. If omitted, OUTPUT is assumed.

KRONREF STATEMENT

The KRONREF control statement generates a cross-reference listing of system symbols used by decks on a Modify OPL.

The control statement format is:

KRONREF(P=lfn_1,L=lfn_2,S=lfn_3,G=lfn_4)

P=lfn_1 — OPL input from file lfn_1. If the P option is omitted or P alone is specified, file OPL is assumed.

L=lfn_2 — List output on file lfn_2. If the L option is omitted or L alone is specified, file OUTPUT is assumed.

S=lfn_3 — System text from overlay lfn_3. The system text must contain symbol definitions. If the S option is omitted or S alone is specified, file NOSTEXT is assumed. If S=0 is specified, only the common deck references and statistics are listed.

G=lfn_4 — System text from local file lfn_4. The system text must contain symbol definitions. 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 central memory) defined in lfn_3 or lfn_4.
- Central memory locations (in low central memory) defined in lfn_3 or lfn_4.
- Control point area words defined in lfn_3 or lfn_4.
- Dayfile message options.
- File types and mass storage constants.
- Job origin types, queue types, and priorities.
- Error flags referenced.
- Common deck calls.
- PP packages called.
- Special entry points.

MODIFY STATEMENT

The MODIFY control statement edits a Modify-formatted program library file.

The control statement format is:

MODIFY(p_1,p_2,...,p_n)

The following optional parameters can be specified in any order.

p_i		Meaning
A	Compressed compile file.	
	Omitted	Write the compile file in uncompressed format.
	A	Write the compile file in compressed format.
C	Compile file output.	
	Omitted or C	Write compile output on file COMPILE.
	C=filename	Write compile output on named file.
	C=0	Write no compile output.
CB	COMPASS binary; meaningful only if Q or X parameter specified.	
	Omitted or CB	Write COMPASS binary output on the load-and-go file (B=LGO).
	CB=filename	Write COMPASS binary output on the named file (B=filename).
	CB=0	Write no binary output (B=0).
CG	COMPASS text retrieval; meaningful only if Q or X parameter specified (takes precedence over CS).	
	CG	Load system text from SYSTEXT (G=SYSTEXT).
	CG=filename	Load system text from named file (G=filename).
	CG=0	SYSTEXT not defined (G=0).
	Omitted	Load system text from overlay named in CS option.
CL	COMPASS list output including *comment lines; meaningful only if Q or X parameter specified.	
	CL	List full listing on OUTPUT file (L=OUTPUT).
	CL=filename	List full listing on named file (L=filename).
	Omitted or CL=0	List short listing on OUTPUT file (L=0).

p_i	Meaning
CS	COMPASS system text; meaningful only if Q or X parameter specified.
	Omitted or CS — Take system text from SYSTEXT overlay (S=SYSTEXT).
	CS=filename — Take system text from named file (S=filename).
	CS=0 — Take no system text (S=0).
CV	Character set conversion.
	Omitted or CV=0 — No conversion takes place.
	CV=63 — Convert library created using 64-character set to 63-character set.
	CV=64 — Convert library created using 63-character set to 64-character set.

NOTE

When the CV=63 or CV=64 parameter is specified, Modify prevents compile file generation (C=0).

D	Debug.
	Omitted — Abort the job on a directive or fatal error.
	D — Do not abort the job on a directive error; the D option does not affect fatal error processing.
F	Full edit.
	Omitted — Decks to be edited are determined by the U parameter or by EDIT directives.
	F — All decks on the library are to be edited and written on new program library, compile file, and source file if the respective options are selected.
I	Directive input.
	Omitted or I — Take directives from job INPUT file.
	I=filename — Take directives from next record on named file.
	I=0 — Do not use directives.

p_i	Meaning	
L	List output.	
	Omitted or L	Write list output on job OUTPUT file. This file is automatically printed.
	L=filename	Write list output on the named file. It is the user's responsibility to assure that the file is saved at job termination or is printed.
	L=0	Do not generate a list output file.
LO	List options.	
	Omitted or LO	Selects list option E if the list output file is assigned to the terminal in a time-sharing job; otherwise, list options E, C, T, M, W, D, and S are selected.
	$LO=c_1c_2...c_n$	Each character (c_i) selects an option to a maximum of seven options. The characters must not be separated.

c_i	Significance
A	Active lines in deck.
C	Directives other than INSERT, DELETE, RESTORE, MODNAME, I, or D.
D	Deck status.
E	Errors.
I	Inactive lines in deck.
M	Modifications performed.
S	Statistics.
T	Text input.
W	Compile file directives.

Example: LO=EMS

N	New program library output.	
	N	Write new program library on file NPL.
	N=filename	Write new program library on named file. It is the user's responsibility to assure that the file is saved at job termination.
	Omitted or N=0	Do not generate a new program library.

NOTE

If a new program library is being generated, an EVICT statement is performed upon it (NPL or filename) before it is written.

p_i		Meaning
NR	No rewind of compile file.	
	Omitted	Rewind compile file at beginning and end of Modify run.
	NR	Do not rewind compile file at beginning or end of Modify run.
P	Program library input.	
	Omitted or P	Take program library input from file OPL.
	P=filename	Take program library from named file.
	P=0	Do not generate program library input file.
Q	Execute named program; no rewind of directives file or list output file.	
	Omitted or Q=0	Do not call assembler or compiler end of the Modify run.
	Q=program	Set LO=E and the A parameter at the beginning of the Modify run. Call the assembler or compiler specified by program at the end of the run.
	Q	Set LO=E and the A parameter at the beginning of the Modify run. Call the COMPASS assembler at the end of the run. When this option is selected, the CB, CL, CS, and CG parameters are meaningful. Compiler input is assumed to be COMPILE. All other parameters are set by default. If CL is not specified with Q, lines beginning with an asterisk in column 1 are not written to the compile file (compile file directives are processed).
S	Source output; illegal when A, Q, or X are selected.	
	S	Write source output on file SOURCE.
	S=filename	Write source output on named file. It is the user's responsibility to assure that the file is saved at job termination.
	Omitted or S=0	Do not generate a source output file.
U	Update edit.	
	Omitted	Decks to be edited are determined by EDIT statement directives or by the F parameter.
	U	Only decks for which directives file contains deck directives are edited and written on the compile file, new program library, and source file (if the respective options are on). F, if specified, takes precedence.
X	Execute named program; directives file and list output file rewound.	
	X	Same as Q option, except Modify directives input (I parameter) and list output (L parameter) files are rewound before processing.

p_i	Meaning
Z	The MODIFY control statement contains the input directives following the terminator; the input file is not read. This eliminates the need for a separate input file for the directives when only a few directives are needed. The first character following the control statement terminator is the separator character for all directives on the control statement. Any display code character that is not used in any of the directives, including a space, can be used as the separator character. The directives can extend to column 72 on the statement. Continuation lines are not permitted. If Z is omitted, the control statement does not contain the input directives.

NOTE

Do not place another terminator after the directive.

For a detailed description of the Modify utility, 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 lnf_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.
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=$c_1c_2...c_n$	Selects up to five c_i list options.

c_i	Description
E	Errors.
C	Input directives.
M	Modifications made.
D	Deck status.
S	Statistics.

If this parameter is omitted or just LO is specified, selects all list options. If the output file is assigned to the terminal in a time-sharing job and this parameter is omitted or just LO is specified, selects list option E.

F	Modify all decks.
D	Debug; ignore errors.
U	Generate *EDIT directives for all decks; meaningful only for *PULLMOD executions.
U=0	Generate no *EDIT directives. If the U option is omitted, generate *EDIT directives for common decks. This is meaningful only for *PULLMOD execution.
Z	The OPLEDIT control statement contains the input directives following the terminator; the input file is not read. This eliminates the need to use a separate input file for the directives when only a few directives are needed. The first character following the control statement terminator is the separator character. If Z is omitted, the control statement does not contain the input directives.

NOTE

Do not place another terminator after the directives.

For a complete description of the OPLEDIT utility, refer to the Modify Reference Manual.

PROFILE STATEMENT

The PROFILE control statement enables the master user to update and inquire about a project profile file for user profile control. Other capabilities of PROFILE (available only to system origin jobs) are described in the NOS System Maintenance Reference Manual.

The control statement format is:

PROFILE(p_1,p_2,...,p_n)

p_i Any of the following in any order:

I=lfn_1 File lfn_1 contains input directives for an update (OP=U). If omitted, file INPUT is assumed.

L=lfn_2 File lfn_2 receives output listings. If omitted, OUTPUT is assumed.

P=lfn_3 File lfn_3 is the project profile file. If omitted, file PROFILB is assumed.

CN=cnum Charge number inquiry. All project numbers valid for charge number cnum are written to output file. Valid only if OP=I option specified.

PN=pnum Project number inquiry. The control values and all valid user numbers for project number pnum are written to the output file. The OP=I and CN=cnum options must also be specified to use the PN option.

CV Convert option. Specifies that directives on the input file are in NOS 1.0 or 1.1 format and are to be converted to update an NOS 1.2, 1.3, or 1.4 profile file. Obsolete directives are ignored. OP=U or OP=T option must also be specified.

OP=optn PROFILE processing option. One of the following:

optn	Description
L	Lists portions of the profile file as specified by the LO parameter.
U	Updates the project profile file with directives supplied by the input file. U is the default value for the master user if the OP option is omitted.
T	Time-sharing update. Processing is the same as OP=U, but preliminary instructions are suppressed.
I	Inquire option. Output is dependent upon CN and/or PN options specified.

LO=*l*sop PROFILE list option (OP=L must be specified). One of the following:

*l*sop	Description
FM	Full list of everything accessible on the profile file by the master user. This is the default for a nonsystem origin job if the LO parameter is omitted.
CM	Charge number list of all charge numbers accessible on the profile file by the master user.
PM	Project number list of all project numbers accessible on the profile file by the master user.

Directives are available to the master user as input to PROFILE to add or update information concerning each charge number. The input file for a PROFILE update (OP=U) is divided into groups of entries that each begin with a charge number directive. Each directive following a particular charge number entry applies only to that charge number, until another charge number entry occurs.

Each line of the input file can contain one or more directives (up to 72 characters per line) in the following format.

$dir_1,dir_2,...,dir_n$

Each directive is separated by a delimiter which can be any special character (display code greater than 44_8) except the following:

/ + - * :

An end-of-line or end-of-card also delimits directives. The following directives are available to the master user for PROFILE input.

dir_i	Description
/cnum	Specifies the charge number cnum to which the following directives apply. If this form of charge number specification is used (refer to CN=cnum directive), it must begin in column 1 with the slash (/).
APN=pnum	Adds or activates project number pnum.
AUN=un	Adds user number un (must be preceded by PN directive).
CN=cnum	Same as /cnum except that it can begin in any column.
DPN=pnum	Deactivates project number pnum. This directive does not delete the specified project number entry but sets its status such that it cannot be specified by users.
DUN=un	Deletes user number un from the list of those who may access the project number (must be preceded by PN directive).
ISV=x	Sets x as the maximum SRU accumulation for any job using the charge number and project number specified by preceding CN and PN directives.
PEX=yymmdd.	Specifies expiration date for project number of preceding PN directive. If PEX=0 is entered, the project number is not limited by an expiration date.
PN=pnum	Project number for which the following directives (until the next PN directive) apply.
SMA=acc	Sets the current number of accumulated SRUs the project number has used (PN directive required). This accumulator is updated at the end of a job or terminal session and each time a CHARGE control statement is entered. When the SMA value surpasses the SML value, the project is not available to users until either the limit or accumulator is respecified.
SML=lim	Specifies the maximum number of accumulated SRUs the project (PN directive) may use. SML=0 implies no restriction.
TI=ti	Specifies the time of day before which the project number specified by the PN directive cannot be used. The time is specified in 24-hour clock notation. For example, a ti specification of 1315 indicates the project number cannot be used before 1:15 in the afternoon.
TO=to	Specifies the time of day before which the project number specified by the PN directive cannot be used. The time is specified in 24-hour clock notation. For example, a ti specification of 1315 indicates that the project number cannot be used before 1:15 in the afternoon.

UPDATE STATEMENT

The UPDATE control statement creates, edits, or copies an Update-formatted program library file. The UPDATE statement is processed in product set format (refer to section 5). For complete information on Update directives and processing, refer to the Update Reference Manual.

The control statement format is:

UPDATE(p_1,p_2,...,p_n)

The following parameters can be specified in any order.

p_i	Description
A	Copies the old sequential access program library to a new random access program library.
B	Copies the old random access program library to a new sequential access program library.
C=lfn_1	Writes compile file decks on file lfn_1. Update writes the compile file using the same character code that was used in the old program library. C=PUNCH writes the decks on file PUNCH and implies the D and 8 parameters. If C=0 is specified, the compile file is suppressed. If lfn_1 or C=lfn_1 is omitted, the compile file is written on file COMPILE. If the K parameter is specified, the C parameter is ignored.
C6=lfn_1	Writes display code compile file decks on file lfn_1. If lfn_1 is omitted, the compile file is written on file COMPILE. If C6=lfn_1 is omitted, the compile file is determined by the C parameter. Do not specify both the C and C6 parameters.
D	Writes 80-column lines on the compile file. If D is omitted, 72-column lines are written.
E	Edits the old program library. To completely edit the library, the E parameter must be specified on two UPDATE control statements. The first occurrence of the E parameter causes the system to rearrange the directory to reflect the actual order of the decks on the library and to remove previously purged identifiers. The second occurrence of the E parameter causes the system to remove the identifiers that exist only as directory entries.
F	Specifies full update mode.
G=lfn_2	Writes PULLMOD directive output on file lfn_2 using the same character code that was used in the old program library. If G=lfn_2 is omitted, PULLMOD directive output is appended to the source file.
G6=lfn_2	Writes PULLMOD directive output in display code on file lfn_2. The user should specify G6=lfn_2 only when OLDPL is in 12-bit ASCII code. If G6=lfn_2 is omitted, the G parameter determines PULLMOD directive output. Do not specify both G and G6 parameters.

p_i	Description

H=n

Specifies the character set used in the Update run.

n	Meaning
3	63-character set.
4	64-character set.
omitted	Character set that was used in the old program library.

$I=lfn_3$

Specifies primary input file lfn_3. If lfn_3 or $I=lfn_3$ is omitted, directives and text are on file INPUT. Directives and text are assumed to be in display code.

$K=lfn_1$

Writes compile file decks on file lfn_1 in the order specified on COMPILE directives. Update writes the decks using the same character code that was used in the old program library. If lfn_1 is omitted, compile file decks are written on COMPILE. If $K=lfn_1$ is omitted, the compile file is determined by the K6, C, or C6 parameters.

$K6=lfn_1$

Writes display code compile file decks on file lfn_1 in the order specified on COMPILE directives. If lfn_1 is omitted, display code compile file decks are written on COMPILE. If $K6=lfn_1$ is omitted, the compile file is determined by the K, C, or C6 parameters. Do not specify both the K and the K6 parameters.

$L=c_1c_2...c_n$

Specifies one or more of the following list options.

c_1	List Options
A	List deck names and correction set identifiers, COMDECK directives, definitions, and decks written on the compile file.
F	All list options except 0.
0	All listing is suppressed.
1	Lists cards in error.
2	Lists active Update directives.
3	Comments on each card that changed status during current execution.
4	Lists text cards.
5	Lists active compile file directives.
6	Lists active and inactive cards.
7	Lists all active cards.
8	Lists all inactive cards.
9	Lists correction history of all cards selected by list options 5, 7, and 8.

P_i	Description
$M=lfn_4$	Specifies program library file lfn_4 to be merged with the old program library. If lfn_4 is omitted, file MERGE is assumed. If $M=lfn_4$ is omitted, no merge program library exists.
$N=lfn_5$	Writes the new program library on file lfn_5. Update writes the new program library using the same character code that was used in the old program library. If lfn_5 is omitted, the new program library is written on NEWPL. If $N=lfn_5$ is omitted, a new program library is not generated for a correction run; creation and copy runs generate a new program library on NEWPL.
$N6=lfn_5$	Writes the new program library in display code on file lfn_5. The user should specify the $N6=lfn_5$ parameter only when OLDPL is in 12-bit ASCII code. If lfn_5 is omitted, the new program library is written in display code on NEWPL. If $N6=lfn_5$ is omitted, the new program library file is determined by the N or N8 parameters.
$N8=lfn_5$	Writes the new program library in 12-bit ASCII code on file lfn_5. If lfn_5 is omitted, the new program library is written in 12-bit ASCII code on NEWPL. If $N8=lfn_5$ is omitted, the new program library file is determined by the N or N6 parameters. Specify only one new program library parameter (N, N6, or N8).
$O=lfn_6$	Writes output on file lfn_6. Update writes the output file using the same character code that was used in the old program library. If lfn_6 is omitted, the output is written on OUTPUT.
$O8=lfn_6$	Writes output in 12-bit ASCII code on lfn_6. If lfn_6 is omitted, display code output is written on OUTPUT. If $O8=lfn_6$ is omitted, the output file is determined by the O parameter. Do not specify both the O and O8 parameters.
$P=lfn_7/s_1/s_2/.../s_7$	Specifies file lfn_7 as the old program library. If just P is specified, file OLDPL is assumed. If $s_1/s_2/.../s_7$ is specified, secondary old program libraries will reside on files $s_1,s_2,...,s_7$.
Q	Process only decks on COMPILE directives (quick mode).
$R=c_1c_2...c_4$	Rewind specified files before and after the Update run.

d_i	Files
C	Compile file.
N	New program library.
P	Old program library and merge library.
S	Source and PULLMOD files.

If c_i is omitted, no files are rewound. If $R=c_1c_2...c_4$ is omitted, the old program library, new program library, compile file, source file, and PULLMOD file are rewound.

p_i	Description
S=lfn_8	Writes source file output on lfn_8 using the same character code that was used in the old program library. If lfn_8 is omitted, source output is written on SOURCE. If S=lfn_8 is omitted, source output is suppressed unless selected by the S6, T, or T6 parameters.
S6=lfn_8	Writes display code source file output on lfn_8. Specify the S6=lfn_8 parameter only when OLDPL is in 12-bit ASCII code. If lfn_8 is omitted, display code source output is written on SOURCE. If S6=lfn_8 is omitted, source output is suppressed unless selected by the S, T, or T6 parameters.
T=lfn_8	Writes source file output on lfn_8 with common decks excluded. Update writes the source file using the same character code that was used in the old program library. If lfn_8 is omitted, source output is written on SOURCE with common decks excluded. If T=lfn_8 is omitted, source output is suppressed unless selected by the S, S6, or T6 parameters.
T6=lfn_8	Writes display code source file output on lfn_8 with common decks excluded. Specify T6=lfn_8 only when OLDPL is in 12-bit ASCII code. If lfn_8 is omitted, display code source output with common decks excluded is written on SOURCE. If T6=lfn_8 is omitted, source file output is suppressed unless selected by the S, S6, or T parameters.
U	Specifies that a fatal error does not terminate Update execution. If U is omitted, a fatal error terminates Update execution.
W	The new program library is written in sequential access format. If W is omitted, the new program library is written in random access format unless the file is on magnetic tape.
X	Writes the compile file in compressed format. If X is omitted, the compile file is not compressed.
8	Writes compile file output as 80-column card images. If 8 is omitted, the compile file is written as 90-column card images.
*=char	The master control character (first character of each directive) for this Update run is char which can be any character having a display code octal value in the range 01 through 54 except for 51 and 52 (the open and close parentheses). If this option is omitted, the master control character is *.
/=char	The comment control character for this Update run is char which can be A through Z, 0 through 9, or +-*/$=. The character should not be changed to one of the abbreviated forms of the directives unless NOABBREV is in effect. If this option is omitted, the comment control character is a slant bar.

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,...,p_n)

p_i Any of the following in any order:

P	Update program library from file OLDPL. If the P option is omitted, file OLDPL is assumed.
P=lfn_1	Update program library from file lfn_1.
N	Modify program library on file OPL.
N=lfn_2	Modify program library on file lfn_2.
M	Modify program library name is OPL. If the M option is omitted, file OPL is assumed.
M=lfn_3	Modify program library name is lfn_3.
F	Convert to file mark.
NR	Do not rewind file lfn_1.

The Update file must be in sequential format. A random Update file must first be changed to sequential format via Update before being submitted to UPMOD for conversion. Unless otherwise specified, only one record from the Update file is converted. After the Modify OPL has been created, no references should be made to modset identifiers present on the Update library. The new OPL should be treated as any other program library created by a Modify creation run.

XEDIT STATEMENT

The XEDIT control statement calls the text editor, XEDIT. For a complete description of XEDIT parameters and commands, refer to the XEDIT Reference Manual.

The control statement format is:

XEDIT(lfn_1,p_1,p_2,...,p_n)dcs

All parameters are optional. The following are brief parameter descriptions.

lfn_1 Name of the file to be edited or created. If lfn_1 is omitted (indicated by two separators before other parameters), the primary file is edited.

p_i One or more of the following parameters in any order.

AS Edit lfn_1 in ASCII time-sharing mode. After the XEDIT job step, processing returns to the original mode.

B Process the job as a batch origin job.

C Create a new file lfn_1.

FR Take the first editing command from the first line of lfn_1.

I=lfn_2 Take editing directives from lfn_2. If I is omitted, commands are taken from file INPUT.

I=0 Take all editing directives from the trailing delimited command sequence. (If the FR parameter is also specified, process the delimited command sequence after processing directives from the first line of lfn_1.)

L=lfn_3 Put all XEDIT output on the specified file.

L=0 Suppress all output.

NH Suppress printing of the XEDIT header.

P Retrieve and edit permanent file lfn_1. Direct access files are attached in write mode. If P is omitted, the file lfn_1 is assumed to be a local file.

dcs Delimited command sequence with the following format (; represents the delimiter character).

;$command_1$;$command_2$;...;$command_n$

LIBRARY MAINTENANCE 14

A library is a file containing records that are accessed individually. Library records can be of several types and can be accessed randomly or sequentially.

This section describes library access methods, library record types, and the following control statements and their functions.

CATALOG	Describes the records on a file.
COPYL and COPYLM	Copies an old file to a new file, updating the records from a replacement file.
GTR	Appends records selected from one file to another file.
ITEMIZE	Describes the records on a file.
LIBEDIT	Generates a file containing records from one or more other files. The records may be of several types. LIBEDIT handles a LIBGEN-generated library as a single record.
LIBGEN	Generates a user library; that is, a library of relocatable and capsule records that can be accessed by CYBER Loader.
VFYLIB	Compares the records in two files and lists their differences.

FILE ACCESS METHODS

The methods used to access records within NOS files are sequential access and random access.

To access a record sequentially, NOS rewinds the file to BOI and then reads records until it finds the requested one. To replace, insert, or delete records from a sequential access file, NOS must rewrite the entire file. (Records can be appended at EOI without rewriting the file.)

To access a record randomly, the file must be on mass storage and must have a directory containing the address of each record (figure 1-14-1). The GTR, LIBGEN, and LIBEDIT statements can generate random access directories. Records within a file can then be replaced, inserted, or deleted by rewriting the directory instead of rewriting the entire file. Records within a random access file can also be accessed sequentially.

LIBRARY RECORD TYPES

NOS determines the type and the name of a record from information contained within the record. If the record begins with a prefix table, the record name (if any) is obtained from that table and the type of the record is determined from the first word following the prefix table. Otherwise, the first word of the record determines the type and name (if any) of the record.

Prefix tables exist, unless they have been specifically suppressed, for:

- Assembled or compiled programs.
- System text overlays.

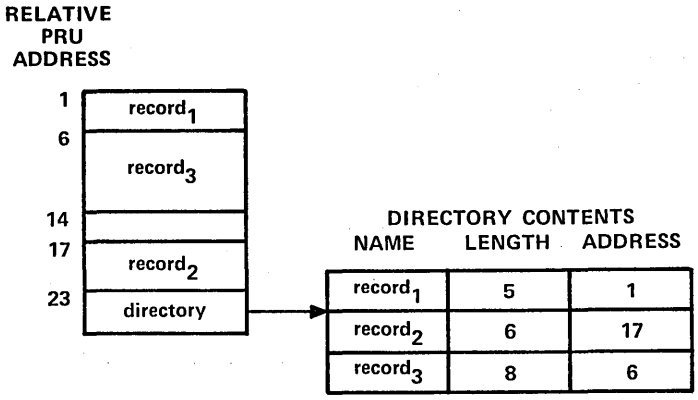


Figure 1-14-1. Random Access File Structure

- User library header records and directory records.
- Modify program library decks, common decks, or directory records.

The prefix table consists of a header word with an octal table type identifier followed by varying amounts of control information.

The prefix table is identified by octal digits 7700 in bits 59 through 48 of its first word; consequently, it is often referred to as a 77 or 7700 table. Information in the prefix table, which originates with the assembler, compiler, or other program that creates the table, can specify items such as the date created and the system on which the job was processed (refer to CYBER Loader Version 1 Reference Manual).

If a prefix table is not present, the first word in a record is examined. If a record meets the criteria for a given type of record, the utilities identify it as such. For instance, a load file beginning with a job statement may be identified as type TEXT, depending on the particular characters in the job statement.

Although some of the records may contain display coded data (loader directives, for instance, are coded), they are considered binary records.

The following record types can be specified on the GTR and LIBEDIT control statements and are recognized by the CATALOG and VFYLIB statements. LIBGEN generates a user library from relocatable (REL) and capsule (CAP) records.

Mnemonic	Meaning
ABS	Multiple entry point overlay.
CAP	CYBER Loader capsule.
OPL	Old program library deck.

Mnemonic	Meaning
OPLC	Old program library common deck.
OPLD	Old program library directory.
OVL	Central processor overlay.
PP	Peripheral processor program.
PPU	Peripheral processor unit program.
PROC	CCL procedure file.
REL	Relocatable central processor program.
TEXT	Unrecognizable as a program.
ULIB	User library directory.

The following record types are recognized by COPYL, COPYLM, and ITEMIZE control statements. Record types ACF, DATA, OPLD, UCF, ULIB, and UPLx are recognized by the ITEMIZE control statement and are not recognized by COPYL or COPYLM.

Mnemonic	Meaning
ABS	Multiple entry point overlay.
ACF	Modify compressed compile file. It has no record name. This record type is determined by nonzero in bits 17 through 0 in the second word of the 77 table.
CAP	CYBER Loader capsule.
DATA	A record that does not have a prefix table or a record name. The first word of the record is binary zero.
OPL	Old program library deck.
OPLC	Old program library common deck.
OPLD	Old program library directory.
OVL	Central processor overlay.
PROC	CCL procedure file.
REL	Relocatable central processor program.
TEXT	A record that has no prefix table. It has a record name in bits 59 through 18 and binary zero in bits 17 through 0.
UCF	Update compressed compile file. It has no record name. This record type is determined by a 77 table with a 0 word count.
ULIB	User library directory.

Mnemonic	Meaning
UPLx	Update sequential program with x master control character. It has no record name. This record type is determined by no 77 table and the characters CHECK in bits 59 through 30; control characters are obtained from bits 5 through 0.
6PP	Peripheral processor program.
7PP	Peripheral processor unit program.

Appendix G in volume 2 contains the formats of PP (6PP), OPL, OPLD, ULIB, and TEXT records. (The OPLC record format is identical to that of the OPL record.) The Modify Reference Manual describes how to create ACF, OPL, OPLC, and OPLD records. The CYBER Loader Reference Manual contains the formats of ABS, CAP, OVL, PP, and REL records. The CYBER Loader Instant contains the PPU (7PP) record format. The Update Reference Manual contains the formats of UCF and UPLx records. Section 4 describes the creation of CCL procedures (PROC).

A user determines the record types contained in a file by issuing a CATALOG statement naming the file. VFYLIB lists the differences between the record types of two files.

CATALOG STATEMENT

The CATALOG control statement lists information about each record in a file assigned to the job. The record types recognized by CATALOG are listed in Library Record Types in this section.

NOTE

CATALOG produces unpredictable results when attempting to catalog an S, L, or F format tape. The user should use the COPY statement to convert the S, L, or F format tape to a mass storage file or to an I or SI binary format tape before attempting to catalog the file.

The control statement format is:

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

lfn Name of the file to be cataloged.

p_i May be any of the following:

N=0 Catalog until an empty file is encountered (two consecutive EOF marks).

N=n Catalog n files; if N=n is omitted, N=1 is assumed.

N Catalog until EOI is encountered.

L=fname Specifies the name of the output file; if L=fname is omitted, CATALOG assumes L=OUTPUT.

U	Lists records within a user library; if U is omitted, only the ULIB record within the user library is listed.
D	Suppresses the comments field; suppresses all page headings after the initial page heading for each file.
CS	Suppresses character set indicator (63 or 64) for OPL and OPLC records.
R	Rewinds lfn before and after cataloging; if R is omitted, lfn is not rewound.

The listing for each file of a multifile file begins on a new page with a page heading for that file. If the D option has been specified, the page heading appears only once, at the beginning of the file. The information listed under each heading is as follows:

Heading	Information
REC	Record number (zero-length records and EOF marks are counted).
NAME	Record name [the contents of the name field from the second word of the prefix (77) table, if present; otherwise, the first seven characters of the record].
TYPE	Record type (refer to Library Record Types in this section).
LENGTH	Record length in words (less prefix table length) printed as an octal number.
CKSUM	A checksum [a value used to verify that the contents of a record (excluding the prefix table) were copied correctly].
DATE	Record creation date (taken from the third word of the prefix table, if present).
COMMENTS	Additional information taken from the prefix table, if present; message terminates before COPYRIGHT comment. (This field is not shown when CATALOG is used in a time-sharing job.)

CATALOG lists additional information depending on the record type. Entry points are listed for REL and ABS records. The character set used, correction identifiers, and their YANK status (refer to the Modify Reference Manual) are listed for OPL and OPLC records.

If the TEXT record name begins with the characters AFRDC, AFRDECK, CMRDC, CMRDECK, DDSDC, DDSDECK, IPRDC, IPRDECK, LIBDC, or LIBDECK, CATALOG lists the entire record. If the TEXT record name begins with OVERLAY, CATALOG lists the first line in the record.

A ULIB record suppresses listing of the other records in the user library unless the U parameter is specified on the control statement.

If an OPLD record in the user library is not encountered before an EOF or EOI within the cataloged file, the following message is output before the *EOF* or *EOI* line.

```
*OPLD MISSING*
```

When a zero-length record is encountered, the length since the last zero-length record is given. If an EOR does not precede an EOF or EOI within the cataloged file, the following message is output before the *EOF* or *EOI* line.

```
*EOR MISSING*
```

The ITEMIZE control statement is similar to the CATALOG statement, but ITEMIZE recognizes additional record types (refer to Library Record Types, earlier in this section).

Example:

Compilation of the FORTRAN program SUBROUT and its subroutines SUB1, SUB2, and SUB3 wrote relocatable object code on file LGO. The following is a catalog of file LGO (refer to the heading definitions given earlier). The I/O file name listed in the FORTRAN PROGRAM statement is flagged by # character.

```
:
.
        CATALOG OF LGO        FILE     1                    80/05/19. 11.55.50.      PAGE      1

     REC   NAME       TYPE        LENGTH     CKSUM     DATE      COMMENTS

       1   SUBROUT    REL            105      5355  80/05/19.    PROGRAMT
S
           SUBROUT
           OUTPUT#
       2   SUB1       REL             45      0263  80/05/19. SUBROUTINET
S
           SUB1
       3   SUB2       REL             45      3276  80/05/19. SUBROUTINET
S
           SUB2
       4   SUB3       REL             45      1542  80/05/19. SUBROUTINET
S
           SUB3
       5   * EOF *        SUM =      264
.
```

COPYL AND COPYLM STATEMENTS

The COPYL and COPYLM control statements copy an old file to a new file, substituting records from a replacement file for the matching records on the old file. Records on the replacement file which do not match records on the old file are either ignored or appended to the new file as specified by the user. Records are considered to match if they have the same type and same name. However, the user may specify that the record type be ignored. COPYL and COPYLM are commonly used to maintain files of procedures or relocatable records.

COPYL and COPYLM differ only in the handling of multiple occurrences of a record on the old file. COPYL uses each record on the replacement file only once, replacing the first matching record from the old file. COPYLM uses the first matching record encountered on the replacement file to replace each matching record from the old file. COPYL can be used to replace multiple occurrences of the same record if multiple occurrences of the record are in the replacement file.

The old file and the replacement file must reside on mass storage or an I or SI format tape. Only a single file terminated by an end-of-file marker is processed by a single call to COPYL or COPYLM unless the user requests processing to the end-of-information by using the E processing option. When working with multifile files, the user must be sure to position the multifile file to the file that is to be processed.

The order of the records on the replacement file is not significant. The system copies the records to the new file in the same order as they are on the old file.

COPYL and COPYLM issue dayfile messages during processing; no other printed output is produced. The dayfile messages list which replacement records were copied and which replacement records were not copied to the new file.

COPYL and COPYLM replace only the types of records listed in Library Record Types, earlier in this section. Any record on the old file that is not recognized as one of the listed types is copied to the new file without further processing. Any replacement file record type that is not listed in Library Record Type is ignored without comment.

The control statement format is:

COPYL(oldlfn,replfn,newlfn,last,flag)	Single replacement.
or	
COPYLM(oldlfn,replfn,newlfn,last,flag)	Multiple replacement.

All parameters are optional and order-dependent. A user denotes an omitted parameter by consecutive commas.

oldlfn	File name of the old file; default name is OLD.
replfn	File name of the replacement file; default name is LGO.
newlfn	File name of the updated file; default name is NEW.
last	Name of the last record on oldlfn to be processed. If last is not specified, all records on oldlfn are processed from its current position to the next EOF (or EOI if the E processing option is used).

flag Processing options.

Flag	Description
R	Rewind oldlfn and newlfn files before processing. (The replfn file is always rewound before and after processing. oldlfn and newlfn are not necessarily rewound to beginning-of-information in multifile files. Refer to explanation below.)
A	Append to the end of newlfn all replfn records that do not match any records on the oldlfn. If A is not selected, records on the replacement file that do not match any records on the oldlfn are ignored and a dayfile message is issued.
T	Check for matching name of record, but omit check for matching type of record. If T is not selected, records match only if both the type and name of the records are the same.
E	Process oldlfn until the end-of-information.

These options can be specified by combining one or more letters in any order, such as TRA, AR, ERTA, or TR.

COPYL and COPYLM check only the first four flag options; if more than four are specified, the remaining characters are ignored.

The R option affects file positioning of the old and new files before processing. If R is specified, the old and new files are rewound before processing. In a multifile file, if there is one or more end-of-file markers between the current position of the file and the beginning-of-information, the R option rewinds the file to the first preceding EOF. In the absence of R, the user is responsible for positioning the oldlfn and newlfn files. The R option does not affect the file of replacement records, since the current file of the replacement file always is rewound to the beginning-of-information before and after processing.

The E option causes the old file to be processed to the end-of-information. Each end-of-file encountered on the old file causes a matching end-of-file to be written on the new file. Records that are added to the new file as a result of combining the A and E options are appended with an end-of-file prior to the end-of-information. In this case, such appended records will follow an end-of-file if both end-of-file and end-of-information existed at the end of the old file.

Processing stops after an end-of-file, end-of-record, or end-of-information is reached, depending on the structure of the old file and the processing options selected. If processing stopped because end-of-file or end-of-record was reached, the old file is positioned after that EOF or EOR. If processing stopped because end-of-information was reached, the old file is positioned just prior to the end-of-information.

COPYL and COPYLM add an end-of-file to the new file even if no end-of-file is encountered on the old file. No further positioning of the new file takes place.

Example:

The following COPYL control statement updates OLDFILE with replacement records from REPFILE and writes them on NEWFILE. The A option appends to the end of NEWFILE any records in REPFILE that do not match a record on OLDFILE.

```
COPYL(OLDFILE,REPFILE,NEWFILE,,A)
```

The following dayfile segment shows that COPYL updated the record named B and appended the record named C.

```
08.53.23.COPYL(OLDFILE,REPFILE,NEWFILE,,A)
08.53.24. UPDATED -- PROC   / B
08.53.24.APPENDED -- PROC   / C
08.53.24. COPYL COMPLETE.
```

The ITEMIZE control statements show the record information for all records in OLDFILE, REPFILE, and NEWFILE.

```
/ITEMIZE(OLDFILE)

1          ITEMIZE OF OLDFILE          FILE      1
    REC    NAME       TYPE          LENGTH    CKSUM        DATE

      1    A          PROC               4     3310
      2    B          PROC               4     5306
      3    (00)            SUM =        10


           * EOI *         SUM =        10

 ITEMIZE COMPLETE.

/ITEMIZE(REPFILE)

1          ITEMIZE OF REPFILE          FILE      1
    REC    NAME       TYPE          LENGTH    CKSUM        DATE

      1    B          PROC               4     1535
      2    C          PROC               4     7304


           * EOI *         SUM =        10

 ITEMIZE COMPLETE.

/ITEMIZE(NEWFILE)

1          ITEMIZE OF NEWFILE          FILE      1
    REC    NAME       TYPE          LENGTH    CKSUM        DATE

      1    A          PROC               4     3310
      2    B          PROC               4     1535
      3    (00)            SUM =        10

      4    C          PROC               4     7304

      5    * EOF *         SUM =        14

 ITEMIZE COMPLETE.
```

GTR STATEMENT

The GTR control statement appends records selected from one file to the end of another file. The records are selected according to directives specifying their type and name. (Refer to Library Record Types in this section for the list of valid record types.) Records can be accessed randomly (default if a directory exists) or sequentially. If specified, a random access directory is appended to the changed file. GTR cannot append records after the directory.

The control statement format is:

GTR(lfn$_1$,lfn$_2$,d,nr,s,na)directive$_1$,directive$_2$,...,directive$_n$

NOTE

The parameters must be in the order shown. GTR identifies its parameters by their position, not by keywords. An omitted parameter is denoted by consecutive commas. Blanks are illegal between the terminator (right parenthesis or period) and directive$_1$.

The parameters are defined as follows:

lfn$_1$ — File which is searched for the requested records; if lfn$_1$ is omitted, file OLD is assumed. lfn$_1$ is always rewound before the GTR operation.

lfn$_2$ — File on which the selected records are written; if lfn$_2$ is omitted, file LGO is assumed. GTR always positions lfn$_2$ at EOI before copying the selected records.

d — Random access directory parameter. If d is specified, lfn$_2$ must be a mass storage file. GTR cannot append records after a directory.

If lfn$_1$ has a random access directory and if sequential access (the S parameter) is not specified, the lfn$_2$ directory record is given the same name as the lfn$_1$ directory record. Otherwise, the lfn$_2$ directory record is named lfn$_2$.

d	Description
Omitted	No new random access directory (OPLD) is added to lfn$_2$. If the directives specify the user library record type (ULIB), the first record of the user library (ULIB) is not copied to lfn$_2$, the relocatable records are copied, and the last record (OPLD) is copied without alteration.
U	No new random access directory (OPLD) is added to lfn$_2$. If the directives specify the user library record type (ULIB), the first record of the user library (ULIB) is copied without alteration to lfn$_2$ along with the relocatable records and the old random access directory (OPLD).

d	Description
D or other	Writes a new random access directory (OPLD) at the end of lfn_2. If the directives specify the user library record type (ULIB), the first record of the user library (ULIB) is copied without alteration to lfn_2 along with the relocatable records and the old random access directory (OPLD).

nr — No rewind option. lfn_1 is not rewound after the operation; lfn_2 is not rewound before or after the operation. If lfn_1 has a directory, the directory is copied to lfn_2.

If nr is omitted, both files are rewound before and after the operation.

s — lfn_1 is searched sequentially; no attempt is made to read a directory.

na — No abort option. If specified, GTR does not search for an EXIT statement when an error occurs. It issues a dayfile message for the error and continues GTR processing at the next directive.

$directive_i$ — Specifies a record or group of records to be retrieved. One or more of the following formats can be used. Valid record types are listed under Library Record Types in this section. The default type is the last type specified on a directive, or if none specified, TEXT. The record name is the first seven characters of the record, or if a prefix table is present, the contents of the name field in its second word.

Directive	Meaning
type/name	Record with the specified type and name.
name	Record with the specified name and the default type.
$type_1/name_1$-$type_2/name_2$	Group of records beginning with $name_1$ of $type_1$ and ending with $name_2$ of $type_2$.
$type_1/name_1$-$name_2$	Group of records beginning with $name_1$ of $type_1$ and ending with $name_2$ of $type_1$.
$name_1$-$name_2$	Group of records beginning with $name_1$ of the default type and ending with $name_2$ of the default type.
type/name-*	All records of the specified type beginning with the named record.
name-*	All records of the default type beginning with the named record.
type/*	All records of the specified type.
*	All records of the default type.
0	A zero-length record is inserted.

GTR searches file lfn_1 for the records specified by the selection directives. If GTR cannot find a record specified by type and name, it issues the following dayfile message.

GTR ERRORS.

If also issues this message when the record specified is within a user library and when the GTR statement syntax is incorrect.

If lfn_1 has a directory (OPLD) record, GTR writes the selected records on lfn_2 in the order specified on the GTR statement. If lfn_2 does not have a directory record, GTR writes the selected records in the order that it finds them on lfn_1, rather than in the order specified on the control statement.

If lfn_2 is on tape, the selected records are copied from the current file position; if lfn_2 is on mass storage, the copy starts at the current EOI of the file. If an EOF exists before the EOI, GTR appends the records as a file following the existing EOF.

Examples:

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

  GTR copies all PP records from file SYSTEM to file BIN. It then builds a random access directory and writes it as the last record on BIN.

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

  GTR retrieves common decks COMCARG and COMCCIO from file OPL. It then writes COMCARG, a zero-length record, and COMCCIO at the current position of file NEW. NEW is not rewound before the operation; OPL and NEW are not rewound following the operation.

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

  GTR copies the user library SYSLIB from file SYSTEM to the end of file SYSLIB.

- GTR.REL/A

  GTR retrieves the relocatable record A from file OLD and copies it to file LGO.

ITEMIZE STATEMENT

ITEMIZE lists pertinent information about each record of a binary file in a format suitable for printing. Earlier in this section, Library Record Type describes the types of records processed by ITEMIZE.

ITEMIZE processes mass storage files or I or SI format tape files. A file can be processed from beginning-of-information through end-of-information.

Output from ITEMIZE is affected by the type of record and parameters selected. A header appears for each file terminated by an end-of-file marker within the file specified by the file name. The first line of the header identifies the file name, file position within that file, and the date and time of the run. The second line of the header has the following fields:

REC	Position of the record within the file.
NAME	Record name obtained from the second word of the prefix table or from the first word of the record.

TYPE	Type of record. Refer to Library Record Type, earlier in this section, for a description of the record types.
LENGTH	Number of words (octal) in the record, excluding the prefix table.
CKSUM	Logical checksum (octal), excluding the prefix table.
DATE	Date record was created as stored in the prefix table.
COMMENTS	Contents of the comments field in the prefix table.

If no prefix table is present, the associated fields are blank.

Additional information listed depends on the type of record:

ABS	Entry point names are listed.
OPL, OPLC, UPL	Deck names are listed.
OVL	Overlay level is listed (octal).
TEXT	Entire record is listed if the name of the record is CMRDC, IPRDECK, IPRDC, LIBDECK, LIBDC, or COMMENT.
6PP	Lists the octal equivalent of the load address.
7PP	Lists the octal equivalent of the load address.

The E parameter can select further details about several types of records (refer to the following control statement description).

The last record in each file is the end-of-file marker, which appears on the listing as the characters *EOF*. The SUM= identification is the total length, in words, for all records in the file, including the prefix table lengths.

Any zero-length record in the file appears with the record name (00). When a zero-length record is encountered, a sum of the lengths of the records encountered since the beginning of the file, or since the last sum was taken, is listed on the output. The length includes prefix tables. Record numbering is not restarted until a new file is encountered.

If a record of type UPL has more correction identifier names and/or deck names than can be accommodated within ITEMIZE, the following message appears on the listing in place of the excess names:

TRUNCATED—IDENT OR DECK LIST TOO LONG

In this instance, the Update utility must be used to obtain a complete list of identifiers and deck names.

A dayfile message is issued when ITEMIZE completes execution.

The control statement format is:

ITEMIZE(lfn,L=listlfn,BL,PW=n,PD,NR,N=n,E)

The first parameter is order-dependent; if lfn is omitted, its position must be indicated by a comma. All the other parameters are optional and order-independent.

Parameter	Description
lfn	Name of file to be itemized; default name is LGO.
L=listlfn	List output on file listlfn; default is L=OUTPUT.
BL	Burstable listing; each file output starts at the top of a page. Default is a compact listing in which a page eject occurs only when the current page is nearly full.
PW=n	Print either 136-character lines or 72-character lines, depending on the value of the decimal integer n. If n ≥136, print 136-character line. If n <136, print 72-character lines. If just PW is specified, print 72-character lines regardless of the listing file device. If PW=n is omitted, the default value is 72-character lines if the listing file is a terminal; otherwise, the default value is 136-character lines.
PD	Print density at eight lines per inch; default is six lines per inch. If this parameter is to produce the desired result, the programmer must ensure that output appears at a printer capable of eight lines per inch.
NR	No rewind of lfn before or after processing; default is rewind before and after processing.
N=n	Itemize n files, where n is a decimal integer; default is N=1. If just N is specified, itemize until end-of-information. If N=0, itemize until an empty file is processed.
E	Expand output to list further information; default is no expansion. For record types CAP and REL, list entry points. For types OPL and OPLC, list modification set names and their YANK status. For record types UPL, list correction identifier names.
U	Itemize all records within ULIB type records; default is list only the user library directory.

If both E and U are selected for ULIB type records, all records in the library are itemized; since the records are all type REL, their entry points are listed.

Example:

A FORTRAN program named SUBROUT has three subroutines. The following ITEMIZE control statement lists the records of the binary object file LGO. The E option lists the entry point for each REL type record listed.

```
/ITEMIZE(LGO,E)

1          ITEMIZE OF LGO              FILE      1
     REC   NAME       TYPE          LENGTH    CKSUM       DATE

       1   SUBROUT    REL              105     5355     08/04/18.
                      SUBROUT    OUTPUT#
       2   SUB1       REL               45     0263     08/04/18.
                      SUB1
       3   SUB2       REL               45     3276     08/04/18.
                      SUB2
       4   SUB3       REL               45     1542     08/04/18.
                      SUB3


           * EOI *          SUM =       264

 ITEMIZE COMPLETE.
```

LIBEDIT STATEMENT

LIBEDIT is a general-purpose utility that generates a file containing records copied from one or more other files (figure 1-14-2). LIBEDIT can build a random access directory for the new file. It recognizes the record types listed in Library Record Types in this section. LIBEDIT processes a user library as a single record.

LIBEDIT can edit a library according to directives requesting addition, deletion, or replacement of specified records from one or more replacement files. Replacement is the implicit mode of a LIBEDIT run. The user must explicitly identify records to be added and records that are not to be replaced (refer to the description of the NOREP directive).

LIBEDIT executes in two phases. During the first phase, it reads directives and replacement records. It groups directives by type and file and groups changes when several insertions take place relative to the same record.

During the second phase, LIBEDIT writes the new file. If LIBEDIT cannot process the specified combination of directives, and the D option (refer to the following control statement description) was not specified, LIBEDIT lists its interpretation of the conflicting directives, issues an error message, and aborts the job step. If the D option was specified, LIBEDIT continues processing the directives.

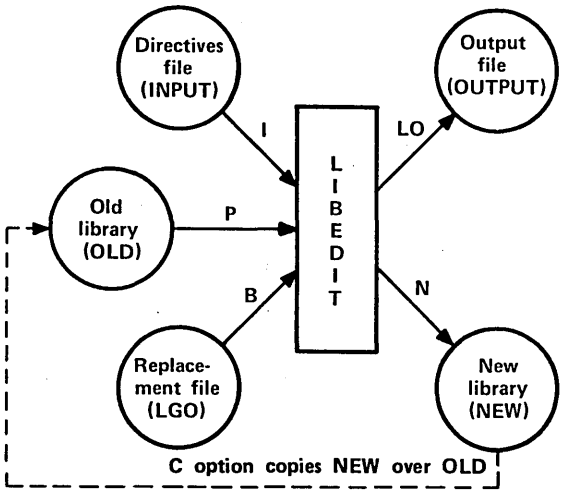


Figure 1-14-2. LIBEDIT Input and Output

CONTROL STATEMENT FORMAT

The following control statement calls LIBEDIT. Its parameters specify options and files used for the call as illustrated in figure 1-14-2.

LIBEDIT(p_1,p_2,...,p_n)

Optional parmeters p_i can be in any order. Each parameter cannot be specified more than once. lfn is a file name of from one to seven alphanumeric characters.

p_i	Meaning
P=lfn_1	Edit file lfn_1 (the old file).
P=0	No old file; new file created from replacement file(s).
P omitted	Edit file OLD.
N=lfn_2	Write new file on file lfn.
N=0	Illegal; error message issued.
N omitted	Write new file on file NEW.

NOTE

The new file is evicted prior to processing (refer to EVICT Statement in section 7).

p_i	Meaning
I=lfn_3	Take directives from the next record of lfn_3.
I=0	No directive input.
I omitted	Take directives from file INPUT.
Z	Take LIBEDIT directives from the control statement line. The directives immediately follow the control statement terminator. The first character following the terminator is the separator character used to separate the directives in the sequence. If can be any character not used in the directives. For example: LIBEDIT,Z.#*DELETE REL/PROG1#*ADD PROG2
Z omitted	Determine the source of directives for the run according to the I parameter specification.
B=lfn_4	Use records from file lfn_4 for insertions and replacements.
B=0	No replacement file used (unless specified by a FILE directive).
B omitted	Use file LGO as a replacement file.
LO=lfn_5	List output on file lfn_5.
LO=0	List no output.
LO omitted	List output on file OUTPUT.
L=1	List directives, modifications, and errors on output file.
L=0	List only errors.
L omitted	Same as L=1.
V	Call VFYLIB after LIBEDIT processing.
V omitted	Do not call VFYLIB.
R	Do not rewind files after processing.
R omitted	Rewind old and new files after processing.
C	Copy the new file over the old file after processing.
C omitted	Do not copy the new file over the old file.
D	Ignore errors and continue.
D omitted	Do not ignore errors; abort job step.

LIBEDIT DIRECTIVES

The user can specify directives to control LIBEDIT processing. These directives can be in a record on file INPUT, on the file specified by the control statement I parameter, or when the Z parameter is specified, as a comment following the control statement.

Directives are not required. If I=0 is specified, LIBEDIT compares the name and type of each record on the old file with those of the records on the replacement file (specified by the B parameter). If a record with the same name and type appears on the replacement file, LIBEDIT writes that record on the new file and skips the record on the old file; otherwise, it copies the record from the old file to the new file. If I=0 and B=0 are specified, LIBEDIT copies the old file to the new file until it encounters an EOF mark or an OPLD directory on the old file.

LIBEDIT recognizes the following directives.

Directive	Function
*ADD	Inserts records before a zero-length record within the file.
*BEFORE or *B	Inserts record before the named record.
*BUILD	Builds a directory at the end of the new file.
*COMMENT	Adds a comment to the prefix table.
*COPY	Copies the new file to the old at the end of editing.
*DATE	Adds the date and a comment to the prefix table.
*DELETE or *D	Does not copy specified records to the new file.
*FILE	Declares a file to be a replacement file.
*IGNORE	Ignores records when reading the replacement file.
*INSERT or *I or *AFTER or *A	Copies record from the replacement file after copying the specified old file record.
*NOREP	Does not automatically replace old file records with records from the specified file.
*RENAME	Renames record.
*REPLACE	Replaces the named records from the old file with records from the replacement file.
*REWIND	Names file to be rewound before and after editing.
*TYPE or *NAME	Sets default record type.

Directive Syntax

A directive begins with an asterisk in column 1 followed immediately by the directive identifier. The directive identifier and the first parameter are both delimited by a comma and/or one or more spaces. If a directive does not begin with an asterisk and a directive identifier, LIBEDIT assumes that the operation is a continuation of the previous directive operation. If an asterisk and directive identifier do not begin the first line of the directives record, LIBEDIT prefixes the following to the first line.

```
*BEFORE*,
```

Parameters other than the first parameter are delimited by a space, an end-of-line, or a comma. A hyphen (-) indicates a record group. Record group identifiers (gid entries) cannot be split between lines. For example, the lines

```
*B,OVL/P1,OVL/P2,OVL/P
3
```

do not constitute a valid directive. The last entry would not be processed as OVL/P3. On the other hand, the lines

```
*B,OVL/P1,OVL/P2
OVL/P3
```

do constitute a valid directive and would be processed as

```
*B,OVL/P1,OVL/P2,OVL/P3
```

Parameters common to many directives are the reference record identifier (rid) and the record group identifier (gid). Valid record types for these parameters are listed in Library Record Types in this section. The default type is the last type specified in a directive; if none are specified, TEXT is the default. The record name is the first seven characters of the record, or if a prefix table is present, the name in its second word. The first character of a record name specified in a directive must not be an asterisk.

rid Reference record identifier specifying the reference point for the requested change. It can have the following formats.

type/name	Reference record has the specified type and name.
name	Reference record has the specified name and is of the default type.
*	Reference point is an end-of-file mark (used with *BEFORE directive only).

gid Record group identifier indicating a record or group of records to be inserted, deleted, or replaced. It can have the following formats.

type/name	Record with the specified type and name.
name	Record with the specified name of the default type.
$type_1/name_1-type_2/name_2$	Group of records beginning with $name_1$ of $type_1$ and ending with $name_2$ of $type_2$.
$type_1/name_1-name_2$	Group of records beginning with $name_1$ of $type_1$ and ending with $name_2$ of $type_1$.

$name_1$-$name_2$	Group of records beginning with $name_1$ of the default type and ending with $name_2$ of the default type.
type/name-*	All records of the specified type beginning with the named record.
name-*	All records of the default type beginning with the named record.
type/*	All records of the specified type.
*	All records of the default type.
0	A zero-length record is inserted.

ADD

The ADD directive inserts records before a zero-length record. A CATALOG listing of the old file numbers each group of records ending with a zero-length record (called a library on the listing). This number on the ADD directive identifies the record group.

NOTE

Adding a zero-length record does not change the directory.

The directive format is:

*ADD LIBn,gid_1,gid_2,...,gid_n

LIBn	Specifies the record group to which the records are appended. Values for n are 1 to 63 and can be determined from a CATALOG listing of the old file.
gid_i	Identifies the records or groups of records from the current replacement file that are to be inserted before the zero-length record.

Example:

The following is a CATALOG listing of file Q.

```
      CATALOG OF Q              FILE     1
REC   NAME      TYPE         LENGTH     CKSUM     DATE

  1   REC1      TEXT              1      5302
  2   REC2      TEXT              1      5304
  3   (00)          SUM =         2   LIBRARY =         1

  4   REC4      TEXT              1      5310

      * EOI *       SUM =         3
```

The following output results when a record was added to file Q, producing file Y.

```
   LIBEDIT DIRECTIVE CARDS.                          78/11/13. 15.12.51.          PAGE     1

       *ADD LIB1,REC3
   RECORDS WRITTEN ON FILE Y                         78/11/13. 15.12.51.          PAGE     2
       RECORD    TYPE      FILE       DATE      COMMENT
       REC1      TEXT      Q
       REC2      TEXT      Q
INSERTED REC3    TEXT      X
       00                  Q

       REC4      TEXT      Q
       **EOF**             Q
```

BEFORE

A BEFORE directive inserts records or groups of records before a specified reference record on the old file. An old file record with the same name and type as an inserted record is not copied to the new file.

The directive formats are:

*BEFORE rid,gid_1,gid_2,...,gid_n

or

*B rid,gid_1,gid_2,...,gid_n

rid — Names the old file record before which the specified replacement file records are to be inserted.

gid_i — Identifies records or groups of records from the current replacement file that are to be inserted before the reference record (rid).

If the first line of the LIBEDIT directives record does not begin with an asterisk and directive name, LIBEDIT assumes that the line is the gid parameters following a *BEFORE*, directive.

BUILD

A BUILD directive constructs and appends a random access directory to the new file. The directory is in Modify format (an OPLD record). If the old file has an OPLD directory, LIBEDIT constructs a directory for the new file with or without a BUILD directive. BUILD can also be used to change the directory name.

The directive format is:

*BUILD dname

dname — One- to seven-alphanumeric-character name for the directory record. No default.

COMMENT

The COMMENT directive adds a comment to the prefix (77) table of a record written on the new file.

The directive format is:

*COMMENT rid comment

rid — Name of a record to be written on the new file.

comment — A string of up to 40 characters that appears in the comment field of the prefix table. Additional characters are truncated.

COPY

The COPY directive directs LIBEDIT to copy the new file over the old file after it has processed all directives.

The directive format is:

*COPY

It performs the same function as the C parameter on the LIBEDIT control statement.

DATE

The DATE directive adds the current date and the specified comment to the prefix (77) table of a record written on the new file.

The directive format is:

*DATE rid comment

rid — Record to be written on the new file.

comment — A string of up to 40 characters to be written in the comment field of the prefix table. Additional characters are truncated.

DELETE

The DELETE directive suppresses copying of the specified records from the old file to the new file.

The directive formats are:

*DELETE $gid_1,gid_2,...,gid_n$

or

*D $gid_1,gid_2,...,gid_n$

gid_i — Identifies records or groups of records that are not to be copied from the old file to the new file.

Example:

```
*DELETE OVL/LAD-REL/RUN
```

This directive requests LIBEDIT not to copy the sequence of records starting with overlay LAD through relocatable CPU program RUN.

FILE

The FILE directive names a file assigned to the job that contains replacement records. LIBEDIT directives following the FILE directive refer to records on the declared replacement file.

The directive format is:

```
*FILE lfn
```

lfn — One- to seven-character name of a relacement file. If lfn is an asterisk (*), LIBEDIT uses the replacement file specified by the LIBEDIT control statement. If the B parameter was omitted from the control statement, LGO is used.

IGNORE

The IGNORE directive requests LIBEDIT to ignore a record or group of records on the current replacement file.

The directive format is:

*IGNORE $gid_1,gid_2,...,gid_n$

gid_i — Identifies records or groups of records on the replacement file that are to be ignored.

Example:

```
*FILE ALPHA
*IGNORE C-*
```

LIBEDIT ignores the sequence of records on file ALPHA starting with record C of the default type and including all records of the default type from C to the EOF mark.

INSERT OR AFTER

The INSERT or AFTER directive requests LIBEDIT to copy the specified records or groups of records from the current replacement file after it has copied the specified old file record onto the new file. Any record on the old file that has the same name and type as an inserted record is not copied to the new file.

The formats for the directives are:

*INSERT rid,gid_1,gid_2,...,gid_n

or

*I rid,gid_1,gid_2,...,gid_n

or

*AFTER rid,gid_1,gid_2,...,gid_n

or

*A rid,gid_1,gid_2,...,gid_n

Example:

*INSERT OPL/K,TEXT/L

This directive requests LIBEDIT to copy the replacement file text record L to the new file after it has copied the old file OPL record K.

NOREP

The NOREP directive declares the specified files to be no-replace files. A no-replace file is a replacement file whose records do not automatically replace old file records having the same name and type. The user selects records to be written on the new file from no-replace files by specifying the file on a FILE directive and then naming the records on *AFTER, *BEFORE, *INSERT, and *REPLACE directives.

The directive format is:

*NOREP lfn_1,lfn_2,...,lfn_n

RENAME

The RENAME directive assigns a new name to a record written on the new file. If the renamed record is referenced by another directive in the directive record, the old name should be used. A RENAME is not allowed on a PROC record.

The directive format is:

*RENAME rid,name

rid Name of the replacement file record or old file record to be renamed.

name One- to seven-alphanumeric-character new name of the record.

REPLACE

The REPLACE directive requests LIBEDIT to replace the old file records having the specified names and types with the replacement file records having matching names and types. This directive is used when the current replacement file has been declared a no-replace file (refer to the NOREP directive description). If the replacement file is not a no-replace file, LIBEDIT performs the replace operation automatically.

The directive format is:

*REPLACE gid_1,gid_2,...,gid_n

gid_i Specifies records or groups of records that appear on both the old file and the current replacement file.

Example:

The old file contains text records A, B, C, and D; the replacement file RF also contains text records named A, B, C, and D. Either of the following directive sequences writes records A and B from the old file and records C and D from file RF onto the new file.

Sequence 1	Sequence 2
*FILE RF	*FILE RF
*NOREP RF	*IGNORE A-B
*REPLACE C-D	

REWIND

The REWIND directive tells LIBEDIT to rewind the specified file before and after processing.

The directive format is:

*REWIND lfn

lfn Name of the file to be rewound.

TYPE OR NAME

A TYPE or NAME directive sets the default record type.

The directive formats are:

*TYPE type
*NAME type

type Specifies default record types. Valid record types are listed in Library Record Types in this section.

The default record type can also be set by an explicit record type specification within a directive. In either case, the default record type setting remains in effect until another record type is explicitly named. If a default record type is not declared in the directive sequence, the default is TEXT. For example, the following two directive sequences are equivalent.

Sequence 1	Sequence 2
*TYPE REL	*INSERT REL/X,Y
*INSERT X,Y	*DELETE FILE1-FILE4
*DELETE FILE1-FILE4	

LIBEDIT OUTPUT

LIBEDIT interprets all directives in the directive record before beginning directive processing. If one or more errors are found, LIBEDIT issues the dayfile message

DIRECTIVE CARD ERROR.

and aborts the job step (unless the D parameter is specified on the control statement). The following LIBEDIT output shows the results of a directive syntax error (the FILE directive is not followed by a space or comma).

```
     LIBEDIT DIRECTIVE CARDS.                          78/11/15. 10.36.47.          PAGE     1
*ERROR*  *FILERF1
```

Directive which cannot be executed are listed as LIBEDIT interpreted them. The following LIBEDIT run called for a replacement file not assigned to the job.

```
     LIBEDIT DIRECTIVE CARDS.                          78/11/15. 10.37.39.          PAGE     1

         *FILE RF1
         *B *,X
*ERROR*  DIRECTIVE CARD CAN NOT BE FOLLOWED.
         *FILE     RF1
         *BEFORE   TEXT/*,TEXT/X
```

Nonfatal errors are listed in an error directory following the listing of records written to the new file. The RECORDS NOT REPLACED error shown in the following example could be corrected by including an *IGNORE directive naming the records not to be replaced.

```
     LIBEDIT DIRECTIVE CARDS.                          79/02/23. 08.37.43.          PAGE     1

         *FILE RF2
         *B *,REC1
     RECORDS WRITTEN ON FILE NEW                       79/02/23. 08.37.43.          PAGE     2
         RECORD    TYPE      FILE       DATE      COMMENT
INSERTED REC1      TEXT      RF2
         **EOF**             OLD

     ERROR DIRECTORY - RECORDS NOT REPLACED.           79/02/23. 08.37.43.          PAGE     3
         RECORD    TYPE      FILE
         REC3      TEXT      RF2
         REC4      TEXT      RF2
```

LIBGEN STATEMENT

The LIBGEN control statement generates a user library of routines for use with CYBER Loader. The control statement format is:

LIBGEN(p_1,p_2,...,p_n)

Any of the following parameters may be specified in any order (only one instance of each).

p_i	Meaning
F=lfn$_1$	Source file containing the relocatable (REL) and/or capsule (CAP) records for the user library. Other record types are ignored.
F or F omitted	LGO is the source file.
P=lfn$_2$	File on which the user library is to be written.
P or P omitted	User library is to be written on file ULIB.
N=name	Name of the user library generated; name entered in ULIB and OPLD records.
N or N omitted	User library name specified by P parameter.
NX=n	If n is not zero, no cross-references are included in the ULIB directory. If n is zero, cross-references are included.
NX or NX omitted	LIBGEN assumes NX=0 (cross-references are included in the directory).

If the F and P parameters specify the same file, LIBGEN issues a dayfile message and does not generate a user library.

Figure 1-14-3 illustrates the structure of a user library. To generate a user library, LIBGEN rewinds and scans the source file, building a directory of all entry points, program names, and external references in the relocatable and capsule records in the file. LIBGEN then copies the source file to the user library file adding the ULIB and OPLD records.

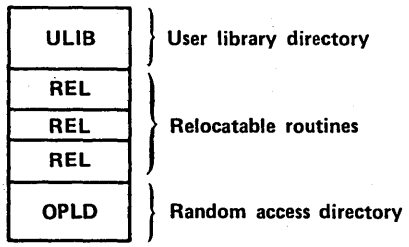


Figure 1-14-3. User Library Structure

Unless the NX parameter specifies otherwise, the ULIB directory contains the external reference/entry point linkage between routines in the user library. When CYBER Loader loads a routine from the user library, it loads (at the same time) all user library routines referenced by the requested routine. All externals for user library routines are satisfied from the user library, if possible. If desired, the user can request with the NX parameter that the ULIB directory contain no cross-linking of records. In that case, when a routine from the user library is requested, only that routine is loaded.

Example 1:

File RELB contains relocatable routines that are used for execution of several applications. To enable loading of these routines as needed during execution of an application program, the user generates a user library using the following control statement.

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

This creates user library APPLIB on file MYLIB. The following loader sequence allows use of the APPLIB routines during execution of a compiled FORTRAN Extended program on file LGO.

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

The program is loaded and executed with externals satisfied first from user library MYLIB and then from the system default library SYSLIB. Refer to the CYBER Loader Reference Manual for more information on library search procedures.

Example 2:

If a routine has no external references, no entry is made in the ULIB directory. To load this routine, the user must include the loader statement LDSET(USEP=pname) in a loader sequence.

Suppose a FORTRAN Extended program contains a BLOCK DATA subroutine without external references to any of its entry points. The user has not named the block, and it has the default name BLKDAT. To load this routine, the user must include the following control statement in the loader sequence.

```
LDSET(USEP=$BLKDAT.$)
```

VFYLIB STATEMENT

The VFYLIB control statement rewinds two files, compares their records, and lists the differences. The record types that VFYLIB recogizes are listed in Library Record Types in this section. VFYLIB lists changes in residence (between record groups separated by zero-length records), replacements, deletions, and insertions. A record is defined as being replaced when its name and type remain the same, but its contents differ. VFYLIB does not compare prefix (77) table information such as last modification date and last assembly date. VFYLIB does not consider a difference in record order as a difference between the two 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, VFYLIB assumes file OLD.
lfn_2	Name of the second file; if this parameter is omitted, VFYLIB assumes file NEW.
lfn_3	Name of the output file; if this parameter is omitted, VFYLIB assumes file OUTPUT.
NR	If specified, lfn_1 and lfn_2 are not rewound after verification.

Example:

The following are CATALOG listings of file OLD and file NEW.

```
     CATALOG OF OLD                FILE     1
REC  NAME      TYPE          LENGTH     CKSUM      DATE
  1  A         REL               25      7547  79/02/28.
       A
  2  (00)           SUM =        25  LIBRARY =         1
  3  B         REL               25      4410  79/02/28.
       B
  4  C         REL               25      1450  79/02/28.
       C
  5  * EOF *        SUM =        77
```

```
     CATALOG OF NEW                FILE     1
REC  NAME      TYPE          LENGTH     CKSUM      DATE
  1  A         REL               30      4122  79/02/28.
       A
  2  B         REL               25      4410  79/02/28.
       B
  3  (00)           SUM =        55  LIBRARY =         1
  4  D         TEXT               1      1000
  5  * EOF *        SUM =        56
```

The control statement, VFYLIB., produces the following listing.

```
VFYLIB.      OLD FILE = OLD      NEW FILE = NEW        79/02/28. 08.58.32.          PAGE     1
     RECORD    TYPE       ULIB          LIB   DATE       COMMENT

RECORDS REPLACED.

     A         REL                       1   79/02/28. 08.56.04   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINEOPT=1

CHANGES IN RESIDENCE.

     B         REL                       1   79/02/28. 08.40.14   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINEOPT=1

DELETED PROGRAMS.

     C         REL                       2   79/02/28. 08.40.14   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINEOPT=1

INSERTED PROGRAMS.

     D         TEXT                      2
```

LIBRARY PROCESSING EXAMPLES

The following examples illustrate the use of CATALOG, GTR, LIBEDIT, and LIBGEN control statements. To duplicate the examples, the user should execute the jobs in sequence.

Example 1:

The following job builds a program library from a replacement file of relocatable binary (REL) records.

```
LIBTES1.
USER(EFD25,PW)
CHARGE(16,13N122)
FTN(L=0)
DEFINE(TESTLIB)
CATALOG(LGO,R)
LIBEDIT(P=0,N=TESTLIB)
CATALOG(TESTLIB,R)
/EOR
      SUBROUTINE A
      STOP
      END
      SUBROUTINE D
      STOP
      END
      SUBROUTINE C
      STOP
      END
      SUBROUTINE B
      STOP
      END
/EOR
*BUILD LIBRARY
*B,*,REL/A,B,C,D
/EOF
```

The FORTRAN Extended compilation produces relocatable binaries on the default file LGO. The DEFINE statement creates a direct access permanent file TESTLIB on which the new program library is written. The first CATALOG statement lists the LGO file as follows:

```
      CATALOG OF LGO              FILE     1                    79/03/01. 08.17.27.      PAGE      1
REC   NAME      TYPE          LENGTH     CKSUM     DATE      COMMENTS
 1    A         REL               25      7547  79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINE
       A
 2    D         REL               25      6705  79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINE
       D
 3    C         REL               25      1450  79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINE
       C
 4    B         REL               25      4410  79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINE
       B

 5    * EOF *       SUM =         124
```

The P=0 in the LIBEDIT statement indicates that no old program library exists. The N parameter indicates that the new program library is written on file TESTLIB. The replacement file is the default LGO. The directives are on the default INPUT file.

LIBEDIT reads the binaries from LGO and the directives from INPUT. On the basis of the directive specifications, the binaries are inserted before the end-of-file on file TESTLIB in the order specified in the directives (A, B, C, D). The directory record created is given the name LIBRARY as a result of the *BUILD directive. It is written before the end-of-file on the new program library TESTLIB.

The directives are written to OUTPUT. The records on file TESTLIB are listed on the next page of OUTPUT. The following listing consists of these two pages.

```
LIBEDIT DIRECTIVE CARDS.                          79/03/01. 08.20.52.          PAGE     1

     *BUILD LIBRARY
     *B, *,REL/A,B,C,D
```

```
     RECORDS WRITTEN ON FILE TESTLIB                          79/03/01. 08.20.52.          PAGE     2
          RECORD     TYPE       FILE       DATE       COMMENT
INSERTED A          REL        LGO        79/03/01. 08.16.29   NOS 1.4  FTN    4.7435  666X  I        SUBROUTINEOPT=1
INSERTED B          REL        LGO        79/03/01. 08.16.29   NOS 1.4  FTN    4.7435  666X  I        SUBROUTINEOPT=1
INSERTED C          REL        LGO        79/03/01. 08.16.29   NOS 1.4  FTN    4.7435  666X  I        SUBROUTINEOPT=1
INSERTED D          REL        LGO        79/03/01. 08.16.29   NOS 1.4  FTN    4.7435  666X  I        SUBROUTINEOPT=1
ADDED    LIBRARY    OPLD       *****      79/03/01.
         **EOF**
```

The second CATALOG statement produces the following listing of information about the records on TESTLIB.

```
     CATALOG OF TESTLIB          FILE     1                          79/03/01. 08.21.45.          PAGE     1
REC  NAME       TYPE         LENGTH     CKSUM     DATE       COMMENTS
  1  A          REL              25      7547  79/03/01. 08.16.29   NOS 1.4  FTN    4.7435  666X  I        SUBROUTINE
       A
  2  B          REL              25      4410  79/03/01. 08.16.29   NOS 1.4  FTN    4.7435  666X  I        SUBROUTINE
       B
  3  C          REL              25      1450  79/03/01. 08.16.29   NOS 1.4  FTN    4.7435  666X  I        SUBROUTINE
       C
  4  D          REL              25      6705  79/03/01. 08.16.29   NOS 1.4  FTN    4.7435  666X  I        SUBROUTINE
       D
  5  LIBRARY    OPLD             13      2073  79/03/01.
  6  * EOF *       SUM =        137
```

Example 2:

This job builds a new program library from an old program library by inserting new relocatable routines into and deleting routines from the old program library created in example 1 (TESTLIB).

```
LIBTES2.
USER(EFD2S,PW)
CHARGE(16,13N122)
FTN(L=0)
ATTACH(OLD=TESTLIB)
DEFINE(NEW=TES2LIB)
LIBEDIT.
CATALOG(NEW,R)
/EOR
        SUBROUTINE BONE
        STOP
        END
        SUBROUTINE D
        STOP
        END
        SUBROUTINE NEWC
        STOP
        END
/EOR
*TYPE REL
*I,B,BONE
*I,C,NEWC
*D,C
/EOF
```

Three relocatable binaries (BONE, D, and NEWC) are produced via a FORTRAN Extended compilation. The old program library (TESTLIB) is attached in read mode and referenced as OLD. A direct access file (TES2LIB) is created for the new program library. This file is referenced as NEW.

LIBEDIT reads the binaries from the replacement file LGO and the input directives from file INPUT. It writes the modified old program library (OLD) to the new program library (NEW). BONE and NEWC are inserted after records B and C, respectively, and record C is deleted. Record D, which already existed on the old program library, is replaced by record D from the replacement file LGO. The following action is taken on file NEW.

```
     LIBEDIT DIRECTIVE CARDS.                          79/03/01. 08.25.52.          PAGE     1

         *TYPE REL
         *I,B,BONE
         *I,C,NEWC
         *D,C
     RECORDS WRITTEN ON FILE NEW                       79/03/01. 08.25.52.          PAGE     2
         RECORD    TYPE      FILE      DATE      COMMENT
         A         REL       OLD       79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINEOPT=1
         B         REL       OLD       79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINEOPT=1
INSERTED BONE      REL       LGO       79/03/01. 08.25.07   NOS 1.4  FTN    4.7435  666X  I       SUBROUTINEOPT=1
DELETED-(C)        REL       OLD
INSERTED NEWC      REL       LGO       79/03/01. 08.25.07   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINEOPT=1
REPLACED D         REL       LGO       79/03/01. 08.25.07   NOS 1.4  FTN    4.7435  666X  I       SUBROUTINEOPT=1
ADDED    LIBRARY   OPLD      *****     79/03/01.
         **EOF**             OLD
```

The CATALOG shows the following contents of the new program library.

```
      CATALOG OF NEW              FILE     1                          79/03/01. 08.26.48.       PAGE       1
REC   NAME        TYPE       . LENGTH     CKSUM     DATE      COMMENTS
  1   A           REL              25      7547  79/03/01..08.16.29   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINE
        A
  2   B           REL              25      4410  79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINE
        B
  3   BONE        REL              25      1103  79/03/01. 08.25.07   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINE
        BONE
  4   NEWC        REL              25      0371  79/03/01. 08.25.07   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINE
        NEWC
  5   D           REL              25      6705  79/03/01. 08.25.07   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINE
        D
  6   LIBRARY     OPLD             15      1312  79/03/01.
  7   * EOF *          SUM =       166
```

Example 3:

This job uses LIBGEN to generate a user library file from the program library file TES2LIB created in example 2.

```
LIBTES3.
USER(EFD25)
CHARGE(16,13N122)
ATTACH(TES2LIB)
DEFINE(LIBFILE)
LIBGEN(F=TES2LIB,P=LIBFILE,N=LOADLIB)
CATALOG(LIBFILE,R,U)
-EOF-
```

The program library TES2LIB is attached to the job. A direct access file LIBFILE 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 LIBFILE, followed by the records from TES2LIB. A file index of addresses for each record in the file is added as the last record of LIBFILE. LOADLIB is the name of the ULIB and OPLD records.

The CATALOG of the user library file LIBFILE shows the following content.

```
      CATALOG OF LIBFILE          FILE     1                          79/03/01. 08.29.02.       PAGE       1
REC   NAME        TYPE         LENGTH     CKSUM     DATE      COMMENTS
  1   LOADLIB     ULIB             13      4267  79/03/01.
  2   A           REL              25      7547  79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINE
        A
  3   B           REL              25      4410  79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINE
        B
  4   BONE        REL              25      1103  79/03/01. 08.25.07   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINE
        BONE
  5   NEWC        REL              25      0371  79/03/01. 08.25.07   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINE
        NEWC
  6   D           REL              25      6705  79/03/01. 08.25.07   NOS 1.4  FTN    4.7485  666X  I        SUBROUTINE
        D
  7   LOADLIB     OPLD             15      6303  79/03/01.
  8   * EOF *          SUM =       201
```

Example 4:

This job illustrates a method for deleting records from a user library. GTR removes the relocatable records (REL) from the user library, LIBEDIT makes the desired changes, and LIBGEN generates a new user library.

```
LIBTES4.
USER(EFD25,PW)
CHARGE(16,13N122)
ATTACH(LIBFILE/M=W)
GTR(LIBFILE,OLD)REL/*
LIBEDIT.
LIBGEN(F=NEW,P=LIBFILE,N=LOADLIB)
CATALOG(LIBFILE,R,U)
-EOR-
*D,REL/NEWC
-EOF-
```

The user library generated in example 4 (LIBFILE) is attached to the job's control point.

Because LIBEDIT handles a user library as a single record, the GTR statement must be used to extract the relocatable records from LIBFILE and write them on the file OLD. (This control statement terminates after OLD; the REL/* is a directive specifying all relocatable records.)

LIBEDIT references the program library OLD and the directive record, deletes NEWC, and writes this modified file on the default NEW. The following is a listing of NEW.

```
   LIBEDIT DIRECTIVE CARDS.                          79/03/01. 08.30.06.          PAGE     1

       *D,REL/NEWC

   RECORDS WRITTEN ON FILE NEW                       79/03/01. 08.30.06.          PAGE     2
       RECORD     TYPE      FILE      DATE      COMMENT
       A          REL       OLD       79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINEOPT=1
       B          REL       OLD       79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINEOPT=1
       BONE       REL       OLD       79/03/01. 08.25.07   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINEOPT=1
DELETED-(NEWC)    REL       OLD
       D          REL       OLD       79/03/01. 08.25.07   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINEOPT=1
       **EOF**               OLD
```

LIBGEN generates a new user library on the file LIBFILE. It uses NEW as the source and names the new user library LIBFILE.

The user library is cataloged, showing the following contents.

```
      CATALOG OF LIBFILE          FILE    1                    79/03/01. 08.30.54.      PAGE     1
REC   NAME       TYPE          LENGTH    CKSUM     DATE      COMMENTS
  1   LOADLIB    ULIB              11     1055  79/03/01.
  2   A          REL               25     7547  79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINE
        A
  3   B          REL               25     4410  79/03/01. 08.16.29   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINE
        B
  4   BONE       REL               25     1103  79/03/01. 08.25.07   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINE
        BONE
  5   D          REL               25     6705  79/03/01. 08.25.07   NOS 1.4  FTN    4.7485  666X  I       SUBROUTINE
        D
  6   LOADLIB    OPLD              13     0414  79/03/01.
```

CHARACTER SETS A

A character set is composed of graphic and/or control characters. A code set is a set of codes used to represent each character within a character set.

A graphic character may be displayed at a terminal or printed by a line printer. Examples are the characters A through Z and the digits 0 through 9. A control character initiates, modifies, or stops a control operation. An example is the backspace character that moves the terminal carriage or cursor back one space. Although a control character is not a graphic character, a terminal may produce a graphic representation when it receives a control character.

All references within this manual to the ASCII character set or the ASCII code set refer to the character set and code set defined in the American National Standard Code for Information Interchange (ASCII, ANSI Standard X3.4-1977). References in this manual to the ASCII character set do not necessarily refer to the ASCII code set.

NOS supports the following character sets.

- CDC graphic 64- (or 63-) character set.
- ASCII 128-character set.
- ASCII graphic 64- (or 63-) character set.
- ASCII graphic 95-character set.

Each installation selects either the 64-character set or the 63-character set. The differences between the two are described in Character Set Anomalies in this appendix. Any reference in this appendix to the 64-character set implies either the 63- or 64-character set, unless otherwise stated.

NOS supports the following code sets.

- Display code.
- 6/12 display code.
- 12-bit ASCII code.

Display code is a set of 6-bit codes from 00_8 to 77_8.

The 6/12 display code is a combination of 6-bit codes and 12-bit codes. The 6-bit codes are 00_8 through 77_8, excluding 74_8 and 76_8. (Refer to Character Set Anomalies for the interpretation of the 00_8 and 63_8 codes.) The 12-bit codes begin with either 74_8 or 76_8 and are followed by a 6-bit code. Thus, 74_8 and 76_8 are considered escape codes and are never used as 6-bit codes within the 6/12 display code set. The 12-bit codes are 7401_8, 7402_8, 7404_8, 7407_8, and 7601_8 through 7677_8. All other 12-bit codes ($74xx_8$ and 7600_8) are undefined.

The 12-bit ASCII code is the ASCII 7-bit code (as defined by ANSI Standard X3.4-1977) right-justified in a 12-bit byte. Assuming that the bits are numbered from the right starting with 0, bits 0 through 6 contain the ASCII code, bits 7 through 10 contain zeros, and bit 11 distinguishes the 12-bit ASCII 0000_8 code from the end-of-line byte. The 12-bit codes are 0001_8 through 0177_8 and 4000_8.

CHARACTER SET ANOMALIES

NOS interprets two codes differently when the installation selects the 63-character set rather than the 64-character set. In tables 1-A-1, 1-A-2, and 1-A-3, the codes for the colon and percent graphic characters in the 64-character set are unshaded; the codes for the colon and percent graphic characters in the 63-character set are shaded.

If an installation uses the 63-character set, the colon graphic character is always represented by a 63_8 code. However, if the installation uses the 64-character set, output of 6/12 display codes 7404_8 or 00_8 produces a colon. In ASCII time-sharing mode, a colon can be input only as a 7404_8 6/12 display code.

When using either the 63- or 64-character set, the use of undefined 6/12 display codes in output files produces unpredictable results and should be avoided.

On input, NOS recognizes alternate 029 punch codes of 11-0 for the right bracket (]) and 12-0 for the left bracket ([). The alternate codes support the COBOL sign overpunch convention and are not recommended for other uses. Refer to the COBOL 4 or COBOL 5 Reference Manual.

Also, two 00_8 codes may be confused with an end-of-line byte and should be avoided (refer to appendix F for further explanation).

CHARACTER SET TABLES

This appendix contains character set tables for time-sharing users, batch users, and magnetic tape users. Table 1-A-1 is for time-sharing users, and table 1-A-2 is for batch users. Table 1-A-3 is a conversion table used to cross-reference 12-bit ASCII codes and 6/12 display codes and to convert ASCII codes from octal to hexadecimal.

Tables 1-A-4, 1-A-5, and 1-A-6 list the magnetic tape codes and their display code equivalents.

The character set tables are designed so that the user can find the character represented by a code (such as in a dump) or find the code that represents a character. To find the character represented by a code, the user looks up the code in the column listing the appropriate code set and then finds the character on that line in the column listing the appropriate character set. To find the code that represents a character, he first looks up the character and then finds the code on the same line in the appropriate column.

TIME-SHARING USERS

Table 1-A-1 shows the character sets and code sets available to an ASCII code terminal user. When in NORMAL time-sharing mode (specified by the NORMAL time-sharing command), NOS displays the ASCII graphic 64-character set and interprets all input and output as display code. When in ASCII time-sharing mode (specified by the ASCII time-sharing command), NOS displays the ASCII 128-character set and interprets all input and output as 6/12 display code.

To determine the octal or hexadecimal ASCII code for a character, refer to table 1-A-3. (Certain terminal definition commands require specification of an ASCII code.)

On output, the US code is reserved for network use and defined as an end-of-line. Use of this character, except in transparent mode, causes incorrect formatting and possible loss of output characters.

BATCH USERS

Table 1-A-2 lists the CDC graphic 64-character set, the ASCII graphic 64-character set, and the ASCII graphic 95-character sets. It also lists the code sets and card punch codes (O26 and O29) that represent the characters.

The 64-character sets use display code as their code set; the 95-character set uses 12-bit ASCII code. The 95-character set is composed of all the characters in the ASCII 128-character set that can be printed at a line printer (refer to Line Printer Usage). Only 12-bit ASCII code files can be printed using the ASCII graphic 95-character set. To print a 6/12 display code file (usually created in time-sharing ASCII mode), the user must convert the file to 12-bit ASCII code. To do this, he issues the FCOPY control statement (section 7). The 95-character set is represented by 12-bit ASCII codes 0040_8 through 0176_8.

LINE PRINTER USAGE

The batch character set printed depends on the print train used on the line printer to which the file is sent (refer to the ROUTE control statement in section 7). The following are the print trains corresponding to each of the batch character sets.

Character Set	Print Train
CDC graphic 64-character set	596-1
ASCII graphic 64-character set	596-5
ASCII graphic 95-character set	596-6

The characters of the default 596-1 print train are listed in the table 1-A-2 column labeled CDC Graphic (64 Char); the 596-5 print train characters are listed in the table 1-A-2 column labeled ASCII Graphic (64 Char); and the 596-6 print train characters are listed in the table 1-A-2 column labeled ASCII Graphic (95 Char).

If a transmission error occurs when printing a line, the system prints the line again. The CDC graphic print train prints a concatenation symbol (↵) in the first printable column of a line containing errors. The ASCII print trains print an underline (_) instead of the concatenation symbol.

If an unprintable character exists in a line (that is, a 12-bit ASCII code outside the range 0040_8 through 0176_8), the number sign (#) appears in the first printable column of a print line, and a space replaces the unprintable character.

To route and correctly print a 6/12 display code file on a line printer with the ASCII graphic 95-character set, a user must convert the 6/12 display code file to a 12-bit ASCII code file with the FCOPY control statement (refer to section 7). The resulting 12-bit ASCII file can be routed to a line printer (refer to the ROUTE statement in section 7) but cannot be output at a time-sharing terminal.

TABLE 1-A-1. TIME-SHARING CHARACTER SETS

ASCII Graphic (64 Char)	ASCII Character (128 Char)	Display Code	6/12 Display Code	12-Bit ASCII Code	ASCII Graphic (64 Char)	ASCII Character (128 Char)	Display Code	6/12 Display Code	12-Bit ASCII Code
: colon †		00†			# num. sign	# num. sign	60	60	0043
Display code 00 is undefined at sites using the 63-character set.					[l. bracket	[l. bracket	61	61	0133
					] r. bracket	] r. bracket	62	62	0135
A	A	01	01	0101	% †	% †	63†	63†	0045
B	B	02	02	0102	: colon	: colon	63	63	0072
C	C	03	03	0103	" quote	" quote	64	64	0042
D	D	04	04	0104	_ underline	_ underline	65	65	0137
E	E	05	05	0105	!	!	66	66	0041
F	F	06	06	0106	& ampersand	& ampersand	67	67	0046
G	G	07	07	0107					
					' apostrophe	' apostrophe	70	70	0047
					?	?	71	71	0077
H	H	10	10	0110	<	<	72	72	0074
I	I	11	11	0111	>	>	73	73	0076
J	J	12	12	0112	@		74		
K	K	13	13	0113	\ rev. slant	\ rev. slant	75	75	0134
L	L	14	14	0114	^ circumflex		76		
M	M	15	15	0115	; semicolon	; semicolon	77	77	0073
N	N	16	16	0116					
O	O	17	17	0117		@		7401	0100
						^ circumflex		7402	0136
						: colon †		7404†	0072
P	P	20	20	0120		X		7404	0045
Q	Q	21	21	0121		` grave accent		7407	0140
R	R	22	22	0122					
S	S	23	23	0123		a		7601	0141
T	T	24	24	0124		b		7602	0142
U	U	25	25	0125		c		7603	0143
V	V	26	26	0126		d		7604	0144
W	W	27	27	0127		e		7605	0145
						f		7606	0146
X	X	30	30	0130		g		7607	0147
Y	Y	31	31	0131					
Z	Z	32	32	0132		h		7610	0150
0	0	33	33	0060		i		7611	0151
1	1	34	34	0061		j		7612	0152
2	2	35	35	0062		k		7613	0153
3	3	36	36	0063		l		7614	0154
4	4	37	37	0064		m		7615	0155
						n		7616	0156
						o		7617	0157
5	5	40	40	0065					
6	6	41	41	0066		p		7620	0160
7	7	42	42	0067		q		7621	0161
8	8	43	43	0070		r		7622	0162
9	9	44	44	0071		s		7623	0163
+	+	45	45	0053		t		7624	0164
-	-	46	46	0055		u		7625	0165
*	*	47	47	0052		v		7626	0166
						w		7627	0167
/	/	50	50	0057		x		7630	0170
(	(	51	51	0050		y		7631	0171
)	)	52	52	0051		z		7632	0172
$	$	53	53	0044		{ left brace		7633	0173
=	=	54	54	0075		| vert. line		7634	0174
space	space	55	55	0040		} right brace		7635	0175
, comma	, comma	56	56	0054		~ tilde		7636	0176
. period	. period	57	57	0056		DEL		7637	0177

† The interpretation of this character or code depends on its context. Refer to Character Set Anomalies in this appendix.

79AAIA
1 OF 2

TABLE 1-A-1. TIME-SHARING CHARACTER SETS (Contd)

ASCII Graphic (64 Char)	ASCII Character (128 Char)	Display Code	6/12 Display Code	12-Bit ASCII Code	ASCII Graphic (64 Char)	ASCII Character (128 Char)	Display Code	6/12 Display Code	12-Bit ASCII Code
	NUL		7640	4000		DLE		7660	0020
	SOH		7641	0001		DC1		7661	0021
	STX		7642	0002		DC2		7662	0022
	ETX		7643	0003		DC3		7663	0023
	EOT		7644	0004		DC4		7664	0024
	ENQ		7645	0005		NAK		7665	0025
	ACK		7646	0006		SYN		7666	0026
	BEL		7647	0007		ETB		7667	0027
	BS		7650	0010		CAN		7670	0030
	HT		7651	0011		EM		7671	0031
	LF		7652	0012		SUB		7672	0032
	VT		7653	0013		ESC		7673	0033
	FF		7654	0014		FS		7674	0034
	CR		7655	0015		GS		7675	0035
	SO		7656	0016		RS		7676	0036
	SI		7657	0017		US†		7677	0037

†Reserved for network use. Refer to Time-Sharing Users in this appendix.

79AAIA
2 OF 2

TABLE 1-A-2. BATCH CHARACTER SETS

CDC Graphic (64 Char)	ASCII Graphic (64 Char)	ASCII Graphic (95 Char)	Display Code	6/12 Display Code	12-Bit ASCII Code	Punch Code	
						026	029
: colon†	: colon†		00†			8-2	8-2
Display code 00 is undefined at sites using the 63-character set.							
A	A	A	01	01	0101	12-1	12-1
B	B	B	02	02	0102	12-2	12-2
C	C	C	03	03	0103	12-3	12-3
D	D	D	04	04	0104	12-4	12-4
E	E	E	05	05	0105	12-5	12-5
F	F	F	06	06	0106	12-6	12-6
G	G	G	07	07	0107	12-7	12-7
H	H	H	10	10	0110	12-8	12-8
I	I	I	11	11	0111	12-9	12-9
J	J	J	12	12	0112	11-1	11-1
K	K	K	13	13	0113	11-2	11-2
L	L	L	14	14	0114	11-3	11-3
M	M	M	15	15	0115	11-4	11-4
N	N	N	16	16	0116	11-5	11-5
O	O	O	17	17	0117	11-6	11-6
P	P	P	20	20	0120	11-7	11-7
Q	Q	Q	21	21	0121	11-8	11-8
R	R	R	22	22	0122	11-9	11-9
S	S	S	23	23	0123	0-2	0-2
T	T	T	24	24	0124	0-3	0-3
U	U	U	25	25	0125	0-4	0-4
V	V	V	26	26	0126	0-5	0-5
W	W	W	27	27	0127	0-6	0-6
X	X	X	30	30	0130	0-7	0-7
Y	Y	Y	31	31	0131	0-8	0-8
Z	Z	Z	32	32	0132	0-9	0-9
0	0	0	33	33	0060	0	0
1	1	1	34	34	0061	1	1
2	2	2	35	35	0062	2	2
3	3	3	36	36	0063	3	3
4	4	4	37	37	0064	4	4
5	5	5	40	40	0065	5	5
6	6	6	41	41	0066	6	6
7	7	7	42	42	0067	7	7
8	8	8	43	43	0070	8	8
9	9	9	44	44	0071	9	9
+	+	+	45	45	0053	12	12-8-6
-	-	-	46	46	0055	11	11
*	*	*	47	47	0052	11-8-4	11-8-4

† The interpretation of this character or code depends on its context. Refer to Character Set Anomalies in this appendix.

79AA2A
1 OF 3

TABLE 1-A-2. BATCH CHARACTER SETS (Contd)

CDC Graphic (64 Char)	ASCII Graphic (64 Char)	ASCII Graphic (95 Char)	Display Code	6/12 Display Code	12-Bit ASCII Code	Punch Code	
						026	029
/	/	/	50	50	0057	0-1	0-1
(	(	(	51	51	0050	0-8-4	12-8-5
)	)	)	52	52	0051	12-8-4	11-8-5
$	$	$	53	53	0044	11-8-3	11-8-3
=	=	=	54	54	0075	8-3	8-6
space	space	space	55	55	0040	no punch	no punch
, comma	, comma	, comma	56	56	0054	0-8-3	0-8-3
. period	. period	. period	57	57	0056	12-8-3	12-8-3
≡ equiv.	# num. sign	# num. sign	60	60	0043	0-8-6	8-3
[l. bracket	[l. bracket	[l. bracket	61	61	0133	8-7	12-8-2†
] r. bracket	] r. bracket	] r. bracket	62	62	0135	0-8-2	11-8-2†
% †	% †	% †	63 †	63 †	0045	8-6	0-8-4
: colon	: colon	: colon	63	63	0072	8-2	8-2
≠	" quote	" quote	64	64	0042	8-4	8-7
↱	_ underline	_ underline	65	65	0137	0-8-5	0-8-5
∨	!	!	66	66	0041	11-0	12-8-7
∧	& ampersand	& ampersand	67	67	0046	0-8-7	12
↑	' apostrophe	' apostrophe	70	70	0047	11-8-5	8-5
↓	?	?	71	71	0077	11-8-6	0-8-7
<	<	<	72	72	0074	12-0	12-8-4
>	>	>	73	73	0076	11-8-7	0-8-6
≤	@		74			8-5	8-4
≥	\ rev. slant	\ rev. slant	75	75	0134	12-8-5	0-8-2
¬	^ circumflex		76			12-8-6	11-8-7
; semicolon	; semicolon	; semicolon	77	77	0073	12-8-7	11-8-6
		@		7401	0100		
		^ circumflex		7402	0136		
		: colon †		7404 †	0072		
		%		7404	0045		
		` grave accent		7407	0140		
		a		7601	0141		
		b		7602	0142		
		c		7603	0143		
		d		7604	0144		
		e		7605	0145		
		f		7606	0146		
		g		7607	0147		

† The interpretation of this character or code depends on its context. Refer to Character Set Anomalies in this appendix.

79AA2A
2 OF 3

TABLE 1-A-2. BATCH CHARACTER SETS (Contd)

CDC Graphic (64 Char)	ASCII Graphic (64 Char)	ASCII Graphic (95 Char)	Display Code	6/12 Display Code	12-Bit ASCII Code	Punch Code	
						026	029
		h		7610	0150		
		i		7611	0151		
		j		7612	0152		
		k		7613	0153		
		l		7614	0154		
		m		7615	0155		
		n		7616	0156		
		o		7617	0157		
		p		7620	0160		
		q		7621	0161		
		r		7622	0162		
		s		7623	0163		
		t		7624	0164		
		u		7625	0165		
		v		7626	0166		
		w		7627	0167		
		x		7630	0170		
		y		7631	0171		
		z		7632	0172		
		{ left brace		7633	0173		
		| vert. line		7634	0174		
		} right brace		7635	0175		
		~ tilde		7636	0176		

79AA2A
3 OF 3

TABLE 1-A-3. ASCII TO 6/12 DISPLAY CODE CONVERSION

ASCII Character (128 Char)	12-Bit ASCII Code		6/12 Display Code	ASCII Character (128 Char)	12-Bit ASCII Code		6/12 Display Code
	Octal	Hex			Octal	Hex	
NUL	4000	00	7640	0	0060	30	33
SOH	0001	01	7641	1	0061	31	34
STX	0002	02	7642	2	0062	32	35
ETX	0003	03	7643	3	0063	33	36
EOT	0004	04	7644	4	0064	34	37
ENQ	0005	05	7645	5	0065	35	40
ACK	0006	06	7646	6	0066	36	41
BEL	0007	07	7647	7	0067	37	42
BS	0010	08	7650	8	0070	38	43
HT	0011	09	7651	9	0071	39	44
LF	0012	0A	7652	: colon †	0072	3A	7404 †
VT	0013	0B	7653	: colon	0072	3A	63
FF	0014	0C	7654	; semicolon	0073	3B	77
CR	0015	0D	7655	<	0074	3C	72
SO	0016	0E	7656	=	0075	3D	54
SI	0017	0F	7657	>	0076	3E	73
				?	0077	3F	71
DLE	0020	10	7660				
DC1	0021	11	7661	@	0100	40	7401
DC2	0022	12	7662	A	0101	41	01
DC3	0023	13	7663	B	0102	42	02
DC4	0024	14	7664	C	0103	43	03
NAK	0025	15	7665	D	0104	44	04
SYN	0026	16	7666	E	0105	45	05
ETB	0027	17	7667	F	0106	46	06
				G	0107	47	07
CAN	0030	18	7670				
EM	0031	19	7671	H	0110	48	10
SUB	0032	1A	7672	I	0111	49	11
ESC	0033	1B	7673	J	0112	4A	12
FS	0034	1C	7674	K	0113	4B	13
GS	0035	1D	7675	L	0114	4C	14
RS	0036	1E	7676	M	0115	4D	15
US ††	0037	1F	7677	N	0116	4E	16
				O	0117	4F	17
space	0040	20	55				
!	0041	21	66	P	0120	50	20
" quote	0042	22	64	Q	0121	51	21
# number sign	0043	23	60	R	0122	52	22
$	0044	24	53	S	0123	53	23
% †	0045	25	63 †	T	0124	54	24
%	0045	25	7404	U	0125	55	25
& ampersand	0046	26	67	V	0126	56	26
' apostrophe	0047	27	70	W	0127	57	27
(	0050	28	51	X	0130	58	30
)	0051	29	52	Y	0131	59	31
*	0052	2A	47	Z	0132	5A	32
+	0053	2B	45	[left bracket	0133	5B	61
, comma	0054	2C	56	\ reverse slant	0134	5C	75
-	0055	2D	46	] right bracket	0135	5D	62
. period	0056	2E	57	^ circumflex	0136	5E	7402
/	0057	2F	50	_ underline	0137	5F	65

† The interpretation of this character or code may depend on its context. Refer to Character Set Anomalies elsewhere in this appendix.
†† Reserved for network use. Refer to Time-Sharing Users in this appendix.

79AA3A
1 OF 2

TABLE 1-A-3. ASCII TO 6/12 DISPLAY CODE CONVERSION (Contd)

ASCII Character (128 Char)	12-Bit ASCII Code		6/12 Display Code	ASCII Character (128 Char)	12-Bit ASCII Code		6/12 Display Code
	Octal	Hex			Octal	Hex	
grave accent	0140	60	7407	p	0160	70	7620
a	0141	61	7601	q	0161	71	7621
b	0142	62	7602	r	0162	72	7622
c	0143	63	7603	s	0163	73	7623
d	0144	64	7604	t	0164	74	7624
e	0145	65	7605	u	0165	75	7625
f	0146	66	7606	v	0166	76	7626
g	0147	67	7607	w	0167	77	7627
h	0150	68	7610	x	0170	78	7630
i	0151	69	7611	y	0171	79	7631
j	0152	6A	7612	z	0172	7A	7632
k	0153	6B	7613	{ left brace	0173	7B	7633
l	0154	6C	7614	| vertical line	0174	7C	7634
m	0155	6D	7615	} right brace	0175	7D	7635
n	0156	6E	7616	~ tilde	0176	7E	7636
o	0157	6F	7617	DEL	0177	7F	7637

79AA3A
2 OF 2

MAGNETIC TAPE USERS

Coded data to be copied from mass storage to magnetic tape is assumed to be represented in display code. NOS converts the data to external BCD code when writing a coded seven-track tape and to ASCII or EBCDIC code (as specified on the tape assignment statement) when writing a coded nine-track tape.

Because only 63 characters can be represented in seven-track even parity, one of the 64 display codes is lost in conversion to and from external BCD code. The following shows the differences in conversion depending on the character set (63 or 64) which the system uses. The ASCII character for the specified character code is shown in parentheses. The output arrow shows how the display code changes when it is written on tape in external BCD. The input arrow shows how the external BCD code changes when the tape is read and converted to display code.

63-Character Set

Display Code		External BCD		Display Code
00		16 (%)		00
33 (0)	Output	12 (0)	Input	33 (0)
63 (:)	⟶	12 (0)	⟶	33 (0)

64-Character Set

Display Code		External BCD		Display Code
00 (:)		12 (0)		33 (0)
33 (0)	Output	12 (0)	Input	33 (0)
63 (%)	⟶	16 (%)	⟶	63 (%)

If a lowercase ASCII or EBCDIC code is read from a nine-track coded tape, it is converted to its uppercase 6-bit display code equivalent. To read and write lowercase ASCII or EBCDIC characters, the user must assign the tape in binary mode and perform his own conversion of the binary data.

Tables 1-A-4 and 1-A-5 show the character set conversion for nine-track tapes. Table 1-A-4 lists the conversions to and from the ASCII character code and display code. Table 1-A-5 lists the conversions between the EBCDIC character code and the display code. Table 1-A-6 shows the character set conversions between external BCD and display code for seven-track tapes.

TABLE 1-A-4. NINE-TRACK ASCII CODED TAPE CONVERSION

ASCII				Display Code		ASCII				Display Code	
Code Conversion†		Character and Code Conversion††				Code Conversion†		Character and Code Conversion††			
Code (Hex)	Char	Code (Hex)	Char	ASCII Char	Code (Octal)	Code (Hex)	Char	Code (Hex)	Char	ASCII Char	Code (Octal)
20	space	00	NUL	space	55	3E	>	1E	RS	>	73
21	!	7D	❩	!	66	3F	?	1F	US	?	71
22	"	02	STX	"	64	40	@	60	`	@	74
23	#	03	ETX	#	60	41	A	61	a	A	01
24	$	04	EOT	$	53	42	B	62	b	B	02
25	%	05	ENQ	%	63	43	C	63	c	C	03
25	%	05	ENQ	space†††	55	44	D	64	d	D	04
26	&	06	ACK	&	67	45	E	65	e	E	05
27	'	07	BEL	'	70	46	F	66	f	F	06
28	(	08	BS	(	51	47	G	67	g	G	07
29	)	09	HT	)	52	48	H	68	h	H	10
2A	*	0A	LF	*	47	49	I	69	i	I	11
2B	+	0B	VT	+	45	4A	J	6A	j	J	12
2C	,	0C	FF	,	56	4B	K	6B	k	K	13
2D	-	0D	CR	-	46	4C	L	6C	l	L	14
2E	.	0E	SO	.	57	4D	M	6D	m	M	15
2F	/	0F	SI	/	50	4E	N	6E	n	N	16
30	0	10	DLE	0	33	4F	O	6F	o	O	17
31	1	11	DC1	1	34	50	P	70	p	P	20
32	2	12	DC2	2	35	51	Q	71	q	Q	21
33	3	13	DC3	3	36	52	R	72	r	R	22
34	4	14	DC4	4	37	53	S	73	s	S	23
35	5	15	NAK	5	40	54	T	74	t	T	24
36	6	16	SYN	6	41	55	U	75	u	U	25
37	7	17	ETB	7	42	56	V	76	v	V	26
38	8	18	CAN	8	43	57	W	77	w	W	27
39	9	19	EM	9	44	58	X	78	x	X	30
3A	:	1A	SUB	:	00	59	Y	79	y	Y	31
Display code 00 is undefined at sites using the 63-character set.						5A	Z	7A	z	Z	32
						5B	[	1C	FS	[	61
3A	:	1A	SUB	:	63	5C	\	7C	|	\	75
3B	;	1B	ESC	;	77	5D	]	01	SOH	]	62
3C	<	7B	{	<	72	5E	^	7E	~	^	76
3D	=	1D	GS	=	54	5F	_	7F	DEL	_	65

† When these characters are copied from/to a tape, the characters remain the same and the code changes from/to ASCII to/from display code.

†† These characters do not exist in display code. Therefore, when the characters are copied from a tape, each ASCII character is changed to an alternate display code character. The corresponding codes are also changed. Example: When the system copies a lowercase a, 61_{16}, from tape, it writes an uppercase A, 01_8.

††† A display code space always translates to an ASCII space.

TABLE 1-A-5. NINE-TRACK EBCDIC CODED TAPE CONVERSION

EBCDIC				Display Code		EBCDIC				Display Code	
Code Conversion†		Character and Code Conversion††				Code Conversion†		Character and Code Conversion††			
Code (Hex)	Char	Code (Hex)	Char	ASCII Char	Code (Octal)	Code (Hex)	Char	Code (Hex)	Char	ASCII Char	Code (Octal)
40	space	00	NUL	space	55	C4	D	84	d	D	04
4A	¢	1C	IFS	[	61	C5	E	85	e	E	05
4B	.	0E	SO	.	57	C6	F	86	f	F	06
4C	<	C0	{	<	72	C7	G	87	g	G	07
4D	(	16	BS	(	51	C8	H	88	h	H	10
4E	+	0B	VT	+	45	C9	I	89	i	I	11
4F	|	D0	}	!	66	D1	J	91	j	J	12
50	&	2E	ACK	&	67	D2	K	92	k	K	13
5A	!	01	SOH	]	62	D3	L	93	l	L	14
5B	$	37	EOT	$	53	D4	M	94	m	M	15
5C	*	25	LF	*	47	D5	N	95	n	N	16
5D	)	05	HT	)	52	D6	O	96	o	O	17
5E	;	27	ESC	;	77	D7	P	97	p	P	20
5F	¬	A1	~	^	76	D8	Q	98	q	Q	21
60	-	0D	CR	-	46	D9	R	99	r	R	22
61	/	0F	SI	/	50	E0	\	6A	|	\	75
6B	,	0C	FF	,	56	E2	S	A2	s	S	23
6C	%	2D	ENQ	%	63	E3	T	A3	t	T	24
6C	%	2D	ENQ	space†††	55	E4	U	A4	u	U	25
6D	_	07	DEL	_	65	E5	V	A5	v	V	26
6E	>	1E	IRS	>	73	E6	W	A6	w	W	27
6F	?	1F	IUS	?	71	E7	X	A7	x	X	30
7A	:	3F	SUB	:	00	E8	Y	A8	y	Y	31
Display code 00 is undefined at sites using the 63-character set.						E9	Z	A9	z	Z	32
						F0	0	10	DLE	0	33
7A	:	3F	SUB	:	63	F1	1	11	DC1	1	34
7B	#	03	ETX	#	60	F2	2	12	DC2	2	35
7C	@	79	\	@	74	F3	3	13	TM	3	36
7D	'	2F	BEL	'	70	F4	4	3C	DC4	4	37
7E	=	1D	IGS	=	54	F5	5	3D	NAK	5	40
7F	"	02	STX	"	64	F6	6	32	SYN	6	41
C1	A	81	a	A	01	F7	7	26	ETB	7	42
C2	B	82	b	B	02	F8	8	18	CAN	8	43
C3	C	83	c	C	03	F9	9	19	EM	9	44

† When these characters are copied from/to a tape, the characters remain the same (except EBCDIC codes 4A, 4F, 5A, and 5F) and the code changes from/to EBCDIC to/from display code.

†† These characters do not exist in display code. Therefore, when the characters are copied from a tape, each EBCDIC character is changed to an alternate display code character. The corresponding codes are also changed. Example: When the system copies a lowercase a, 81_{16}, from tape, it writes an uppercase A, 01_8.

††† All EBCDIC codes not listed translate to display code 55_8 (space). A display code space always translates to an EBCDIC space.

TABLE 1-A-6. SEVEN-TRACK CODED TAPE CONVERSION

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

†As explained previously in this section, conversion of these codes depends on whether the tape is being read or written.

MESSAGES B

This appendix contains an alphabetical listing of the messages that may appear in a user's dayfile or output file. Lowercase characters identify variable names or fields. Messages beginning with variable names or characters follow those beginning with A through Z and 0 through 9. These messages are alphabetized according to the first nonvariable word or character. Messages beginning with any special characters (such as hyphens or asterisks) are alphabetized as if the special character were not present. For example, the message

pfn ALREADY PERMANENT, AT nnn.

is listed after the messages beginning with A through Z and 0 through 9 and is alphabetized with the messages whose first nonvariable word or character begins with A.

Dayfile messages usually issued only to COMPASS programs are listed in appendix B in volume 2. If a message received during processing of a time-sharing job is not listed in this appendix or in appendix B in volume 2, refer to the Network Products IAF Reference Manual or the NOS Time-Sharing User's Reference Manual.

If you encounter a diagnostic or informative message that does not appear in this appendix, consult the NOS Diagnostic Index. This publication catalogs all messages produced by NOS and its products and specifies the manual or manuals in which each message is fully documented.

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ACCOUNT BLOCK LIMIT.	The monitor detected the expiration of the account block SRU limit.	Reset account block SRU limit with SETASL control statement or macro. If the account block limit is set at its maximum, issue another CHARGE statement to begin a new account block.	1AJ
ADDRESS OUT OF RANGE addr.	LOC read an address addr on a correction statement that is greater than or equal to the user's field length.	The correction statement is ignored and LOC continues.	CPMEM
ADDRESS OUT OF RANGE.	An address in a parameter block is outside the user's field length.	Specify parameter block address within field length.	LFM
APPENDED - type/name	The record with type and name on the replacement file was not matched; it has been appended to the new file. A option selected.	None.	COPYL, COPYLM
ARG. ERROR.	LDR parameters were outside the user's field length.	Examine program to determine error.	LDR
ARGUMENT ERROR.	A control statement is syntactically incorrect. Refer to the appropriate control statement or command for further information. When the system processes tape management statements, it issues this message if both ring enforcement options (PO=R and PO=W) or more than one EOT option (PO=I, PO=P, and PO=S) is specified. Also, specification of duplicate parameters (more than one occurrence of a keyword) or multiple equivalent parameters (such as MT/NT, CB/CK, FI/L, R/W, and so forth) is not allowed on tape assignment control statements. The address parameter on DMPECS, DMDECS, LBC, LOC, PBC, or WBR must be numeric.	Recheck parameters.	CONVERT, COPY, COPYBR, COPYBF, COPYEI, COPYX, CPMEM, LO72, RESEX, TCOPY, UPMOD
ARGUMENT ERRORS.	The ENQUIRE control statement is syntactically incorrect.	Check parameters on control statement and retry.	ENQUIRE

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ARITHMETIC INDEFINITE.	The CPU floating-point arithmetic unit attempted to use an indefinite operand.	Analyze the job output and dumps to determine the cause.	1AJ
ARITHMETIC OVERFLOW.	The CPU floating-point arithmetic unit received an operand too large for computation.	Analyze the job output and dumps to determine the cause.	1AJ
ARITHMETIC UNDERFLOW.	The CPU floating-point arithmetic unit received an operand too small for computation.	Analyze the job output and dumps to determine the cause.	1AJ
BAD DECK NAME.	A deck name has more than seven characters.	Correct error and rerun.	UPMOD
BAD PARAMETER ON LIBEDIT CARD.	The LIBEDIT control statement contains an incorrect parameter.	Check the LIBEDIT control statement description for the correct statement format.	LIBEDIT
BINARY SEQ. ERROR, RECxxxx CDyyyy.	A binary card was found to be out of sequence and the job was terminated without EXIT processing. Error is on card yyyy (octal) of record xxxx (octal).	Examine sequencing of binary deck and correct error.	1AJ
BLANK LABELS DO NOT VERIFY.	The tape labels written on the blank tape could not be verified, probably due to hardware failure.	Request that the operator blank label the tape using a different tape drive.	BLANK
BLANK TAPE, lfn AT addr.	A blank tape was read. (Blank tape is defined as more than 25 feet of erased tape.)	Ensure correct tape is specified on control statement.	1MT
BLOCK SEQUENCE ERROR, lfn AT addr.	The block length recorded in the file did not match the length of the block read, or the block number recorded in the file did not match the system block count (this message applies to I format tapes only).	Ensure accuracy of format parameter (F) on control statement or macro.	1MT
BLOCK SIZE NOT APPLICABLE.	Either the CC or BS parameter was specified on a COPY control statement for which neither the input nor output file was an S or L format tape or the CC parameter was specified on a TCOPY control statement without the E or B conversion parameter	Refer to the COPY or TCOPY control statement description, correct the statement, and retry.	COPY, TCOPY

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	specified.		
BLOCK SIZE TOO LARGE ON filenam.	S, L, or F tape block size exceeds copy specifications.	Reduce the block size.	COPY, TCOPY
BLOCK SIZE TOO SMALL ON filenam.	Block size on S, L, or F tape does not meet copy requirements.	Increase block size so that it is greater than the noise size. On F to F tape copy, increase output file block size so it is greater than the block size of the input file.	COPY, TCOPY
BLOCK TOO LARGE, lfn AT addr.	The tape being read contained a data block greater in size than that allowed by the specified format or by user declaration.	Ensure accuracy of format parameter (F) on control statement or macro.	1MT
BOT/EOT ENCOUNTERED, lfn AT addr.	Indicates an abnormal tape position.	Inform site analyst if persistent.	1MT
BREAKPOINT CONDITION.	The job executed an address for which a breakpoint was requested by the system.	Inform site analyst.	1AJ
BUFFER ARGUMENT ERROR, lfn AT addr.	For tape operations, this message indicates one of the following. - FET less than 7 words long for S/L format - MLRS greater than 1000 octal for S format - POSMF issued and no HDR1 label found in FET or extended label buffer Refer to volume 2 of the NOS Reference Manual.	Examine program to determine error.	1MT
BUFFER ARGUMENT ERROR ON lfn AT addr.	A buffer pointer did not conform to the following constraints. - FIRST .LE. IN - FIRST .LE. OUT - OUT .LT. LIMIT .LE. FL Refer to volume 2 of the NOS Reference Manual.	Examine program to determine error in buffer pointers.	CIO
BUFFER CONTROL WORD ERROR.	Dayfile message indicating that the word count in the disk linkage is greater than 100B.	Inform site analyst.	SLL
BUFFER CONTROL WORD ERROR, lfn AT addr.	Either an attempt was made to write a block smaller than the noise size on an S, L, or F format tape, or a control word error occurred in a write (such as bad byte	Examine program to determine error.	1MT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	count). Refer to volume 2 of the NOS Reference Manual.		
CATALOG ARGUMENT ERROR.	The syntax of the CATALOG control statement is incorrect.	Check the CATALOG statement description and correct the statement.	CATALOG
CATALOG COMPLETE.	Informative message indicating that cataloging or the list run is complete.	None.	CATALOG, MODVAL, PFATC
CATALOG FILE NAME CONFLICT.	The same file was named as the file to be cataloged and as the file onto which the catalog is written.	Change one of the file names on the CATALOG control statement.	CATALOG
CATALOG OVERFLOW - FILES, AT addr.	The number of files in the user's catalog exceeds his limit.	One or more permanent files must be purged in order to save or define additional files.	PFM
CATALOG OVERFLOW - SIZE AT addr.	The cumulative size of the indirect access files in the user's catalog exceeds his limit.	One or more indirect access files must be purged or shortened to allow additional permanent file space.	PFM
CCL100-SEPARATOR FOLLOWING VERB MUST BE COMMA OR LEFT PARENTHESIS	Fatal user error. Separator following verb in a CCL statement must be a comma or a left parenthesis.	Change separator following verb to a comma or a left parenthesis.	CCL
CCL101-LAST NON-BLANK CHARACTER MUST BE SEPARATOR	Fatal user error. Last character string of card or line was not followed by a separator or a terminator.	To terminate statement, make last nonblank character a period or right parenthesis. To continue statement on next card or line, make last nonblank character a valid separator.	CCL
CCL102-EQUAL SIGN MUST FOLLOW FIRST SYMBOLIC NAME	Fatal user error. First parameter following a SET verb is a symbolic name to be set. An equal sign must follow the symbolic name.	Change separator following the symbolic name to an equal sign. Equal sign must be followed by an expression.	CCL
CCL103-STATEMENT INCOMPLETE	Fatal user error. A terminator was detected immediately following a verb.	Check statement format (refer to section 4 of NOS Reference Manual, volume 1) and rewrite	CCL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
		statement, using a comma or left parenthesis after verb.	
CCL104-EXPRESSION AND STATEMENT TERMINATED BY t	Fatal user error. A CCL expression or verb was followed by a statement terminator (period or right parenthesis) instead of a comma. Any of following conditions may produce this error. This message is informative and is issued in conjunction with message CCL123. - The label string parameter was omitted. - A relational or logical operator was delimited by a period on left side only. - An unbalanced right parenthesis appeared in expression. If separator following verb was a left parenthesis, an unbalanced right parenthesis may appear balanced. For example, IFE(R1+R2)=3, LABEL. is terminated by right parenthesis.	- Place a comma after expression or verb and add label string parameter. - Delimit operator with periods on both sides. - If error was caused by a misleading left parenthesis following verb, add a comma immediately after verb. Otherwise, recheck all parentheses and balance unbalanced right parenthesis.	CCL
CCL105-TERMINATOR MUST FOLLOW SUBSYSTEM NAME	Fatal user error. The SS function must be of the form SS. or SS=name.	Place a period in the appropriate location.	CCL
CCL120-ALL CARDS SKIPPED - xxxxxx	Fatal user error. In process of skipping, end of control statement record was reached. Skipping was initiated by a CCL statement in either control statement file (ENDRUN) or a procedure (REVERT). Any of following conditions may produce this error. - An IFE, SKIP, or ELSE expression was false and no ENDIF statement terminated resultant skip. - A WHILE expression was false and no ENDW statement terminated resultant skip. - An ENDW statement was not preceded by a WHILE statement.	- Either add an ENDIF statement with a matching label string or correct existing ENDIF statement. - Either add an ENDW statement with a matching label string or correct existing ENDW statement so that WHILE and ENDW label strings match. - Precede ENDW statement with a WHILE statement.	CCL
CCL121-LABEL STRING MUST BEGIN WITH ALPHA CHARACTER	Fatal user error. First character of a label string must not be numeric.	Change label string to a label string of 1 to 7 alphanumeric characters, beginning with an alphabetic character.	CCL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
CCL123-ALPHANUMERIC LABEL STRING REQUIRED	Fatal user error. CCL statements ELSE, ENDIF, ENDW, IFE, SKIP, and WHILE require label string parameter. This message is issued in conjunction with message CCL104.	Refer to message CCL104 for cause and suggested action.	CCL
CCL124-PRECEDING ERR MSG. APPLIES TO FOLLOWING	Informative message. Dayfile message which precedes CCL124 is an error message which was caused by the statement printed after CCL124.	Refer to error message which precedes this in the dayfile for cause and suggested action.	CCL
CCL125-NUMBER OF CARDS SUPPRESSED=x	Informative message. A statement which CCL is printing from an internal buffer continued over more cards or lines of input than buffer size. CCL prints first three cards or lines and then prints this message to indicate number of cards or lines suppressed, x. Remaining cards or lines of statement follow this message.	None.	CCL
CCL126-STATEMENT TERMINATOR MUST FOLLOW LABEL	Fatal user error. Either of the following conditions may produce this error. - A label string is last parameter on an ELSE, ENDIF, ENDW, IFE, SKIP, or WHILE statement. It must be followed by a period or a right parenthesis. - A CCL expression contained a comma, and character string that followed was assumed to be a label string.	- Terminate CCL statement with a period or a right parenthesis. - Either remove comma or replace it with a left parenthesis, depending on nature of error.	CCL
CCL127-LABEL STRING EXCEEDS xx CHARACTERS	Fatal user error. Label string parameter on an ELSE, ENDIF, ENDW, IFE, SKIP, or WHILE statement is greater than xx characters. xx is maximum number of characters defined by installation.	Change label string on any CCL statement it appears to a label string with xx or fewer alphanumeric characters.	CCL
CCL150-EXPRESSION CONTAINS INVALID OPERATOR x	Fatal user error. x is not a legal operator in preceding CCL statement.	If x is part of a character string, $-delimit character string. If x is an intended operator, refer to section 4 for a list of valid operators.	CCL
CCL151-EXPRESSION FORMAT ERROR (IN FILE FUNCTION)	Fatal user error. Format of expression is illegal. Probable cause is use of commas or unbalanced left parentheses within an expression. If error was detected within a CCL FILE function, IN FILE FUNCTION is included in message. FILE function must be of form FILE(lfn,exp), where exp is a valid FILE function.	Check parentheses and balance unbalanced left parenthesis or remove excess commas from FILE function.	CCL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
CCL152-POORLY FORMED FUNCTION (NUM, DT, SS)	Fatal user error with following causes. - Separator after CCL function name is not a left parenthesis or function is not terminated by a right parenthesis. - Character string evaluated by CCL function contained special characters but was not $-delimited.	- Correct format of function. - $-delimit character string.	CCL
CCL153-ILLEGAL FUNCTION CALL WITHIN FILE FUNCTION	Fatal user error. A CCL NUM, SS, or FILE function was called from within a FILE function.	Restructure CCL expression so NUM, SS, or FILE function is not within a FILE function.	CCL
CCL154-ILLEGAL EXPONENT	Fatal user error. A negative exponent in a CCL expression is illegal.	Remove minus sign preceding exponent.	CCL
CCL155-OPERATOR OR OPERAND SEQUENCE ERROR	Fatal user error with following causes. - Sequence of adjacent operators is illegal. For example, 3*-4 is not allowed. - An operand is adjacent to a left or right parenthesis. Implied multiplication is not allowed. For example, 3(R+1) is illegal.	- Use parentheses to separate adjacent operators. - Use an operator to separate operand and parenthesis.	CCL
CCL156-STRING TOO LONG -strng	Fatal user error. Listed character string is too long for a CCL expression. A character string may not be longer than 10 characters, excluding $ delimiters for literals.	Shorten character string to 10 characters or less.	CCL
CCL157-UNKNOWN NAME -strng	Fatal user error. Alphanumeric character string strng is not a symbolic name recognized by CCL.	Replace strng with a valid symbolic name. Refer to Section 4 for a list of valid symbolic names.	CCL
CCL158-OPERATOR/TERMINATOR MUST FOLLOW FUNCTION (DT, FILE, NUM)	Fatal user error. CCL function was properly formed and positioned within CCL statement, but it was not followed by an operator, separator, or terminator.	Place an operator, separator, or terminator after function.	CCL
CCL159-FIRST PARAMETER INVALID IN SET STATEMENT	Fatal user error. First parameter in SET statement was not one of following symbolic names: DSC, EF, EFG, R1, R2, R3, R1G.	Replace first parameter with a valid symbolic name.	CCL
CCL160-NUMERIC OR LITERAL IN FILE FUNCTION	Fatal user error. Expression of a FILE function may contain only symbolic names defined for function.	Remove numeric or literal character string. Refer to Section 4 for a list of FILE symbolic names.	CCL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
CCL161-STACK OVERFLOW	Fatal user error. Evaluation of an expression overflowed either operator or operand stack.	Check expression for errors and correct. If no errors, either simplify expression or break it apart so that two or more CCL statements have same effect as one.	CCL
CCL163-SUBSYSTEM REFERENCE ERROR	A name referenced by the SS function does not exist in table of names known to CCL.	Refer to section 4 for a list of valid names.	CCL
CCL200-PROCEDURE NESTING LEVEL xx EXCEEDED	Fatal user error. Current procedure call forced procedure nesting to exceed limit of xx, which is defined by installation.	Reposition procedure call statements so limit xx is not exceeded.	CCL
CCL201-PROCEDURE NAME MORE THAN 7 CHARACTERS	Fatal user error. Procedure name cannot be greater than 7 characters.	Rename procedure.	CCL
CCL203-PROCEDURE FILE NAME NOT SPECIFIED OR INVALID	Fatal user error with following causes. - File name specified on BEGIN statement was greater than 40 characters. - pfile parameter on BEGIN statement was null, indicating default file name. CCL was installed with default file flag turned off; no default file name is allowed.	- Specify a file name with seven or fewer characters. - Specify a file name.	CCL
CCL204-MULTIPLE EQUIVALENCE SPECIFICATIONS FOR xx	Informative message. A keyword has been specified more than once on a procedure call statement. The last specification is used.	If a preceding specification is desired, remove succeeding specifications. Otherwise, no action is required.	CCL
CCL205-FORMAL PARAMETER LIST DOES NOT INCLUDE -x	Fatal user error. While in equivalence mode, CCL discovered a keyword x on the procedure call statement that was not specified on header statement.	Remove keyword x from procedure call statement.	CCL
CCL206-SYMBOLIC SPECIFICATION INVALID xxx+	Fatal user error. A parameter, xxx, on procedure call statement is followed by a plus sign. Plus sign indicates that CCL is to convert value to display code, and is valid only when preceded by a symbolic name and followed by a D, B, or a null field. (Example: R1+B is a legal value, but 37+R1 is not.)	If plus sign is part of a character string, $-delimit character string. If plus sign should convert a symbolic name to display code, replace xxx with a valid symbolic name. Refer to section 4 of NOS Reference Manual,	CCL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
		volume 1 for a list of valid symbolic names.	
CCL207-PROCEDURE NAMED BEGIN IS INVALID	Fatal user error. A procedure must not be named BEGIN.	Select another procedure name.	CCL
CCL211-SPECIFICATION EXCEEDS xx CHARACTERS	Fatal user error. Value on a procedure call statement or default value on a procedure header statement is greater than xx characters. xx is defined by installation.	Specify a value with xx or fewer characters.	CCL
CCL212-SEPARATOR INVALID strng s	Fatal user error. s is illegal separator and strng is character string that precedes s. Any of following conditions produces this error. - On a procedure call statement, the separator preceding a formal keyword or a positionally specified value is not a comma. - Invalid separator is part of a character string. If a valid separator immediately precedes illegal separator s, the string is null. - #DATA or #FILE has been specified on the procedure call statement.	- Change separator s to a comma. - $-delimit character string. - Remove #DATA or #FILE from procedure call statement and specify on procedure header statement.	CCL
CCL230-PROCEDURE FILE NOT FOUND	Fatal user error. The file named on the BEGIN statement was not assigned to the job and the installation had chosen to inhibit automatic retrieval of a permanent file with that name.	Check file name for errors and correct. If name is correct, retrieve file prior to BEGIN statement.	CCL
CCL231-PROCEDURE NOT FOUND	Fatal user error. Local or permanent file indicated on BEGIN statement was found, but CCL could not find procedure on the file.	Check procedure name for errors and correct. If name is correct, check file for procedure.	CCL
CCL234-UNABLE TO LOCATE LIBRARY PARTITION xxx	Fatal user error. CCL could not find procedure xxx, specified on call-by-name statement, on currently defined library set.	Check procedure name for errors and correct. If name is correct, check library for procedure.	CCL
CCL235-FORMAL PARAMETER GT xx CHARACTERS	Fatal user error. Number of characters in a keyword on header statement exceeds xx, as defined by installation.	Define a keyword with xx or fewer characters.	CCL
CCL236-SPECIAL DEFAULT SPECIFICATION UNKNOWN	Fatal user error. Equivalence symbol, which specifies a special default option on procedure header statement, must be followed by a known keyword of either FILE or DATA.	If equivalence symbol is part of a character string, $-delimit character string. If a special default option	CCL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
		is desired, follow equivalence symbol with either DATA or FILE.	
CCL237-SEPARATOR FOLLOWING SECOND DEFAULT IS */*	Fatal user error. In form keyword=default1/ default2/ on procedure header statement, second / is illegal.	If / is part of default2, $-delimit character string. If not, either remove it or replace it with a comma or period.	CCL
CCL238-FORMAL PARAMETER LIMIT xx EXCEEDED	Fatal user error. Number of parameters on procedure header statement has exceeded xx, as defined by installation.	Remove parameters from procedure header statement until xx or fewer parameters remain.	CCL
CCL239-PROCEDURE HEADER NOT TERMINATED	Fatal user error. Procedure header statement was not terminated by a period, and no control statements were found after header statement.	Terminate header statement with a period, and add at least one control statement to procedure.	CCL
CCL249 - DATA FILE LFN EXCEEDS 7 CHARACTERS - lfn	The file name lfn specified on a .DATA command is longer than seven characters.	Specify a shorter file name.	CCL
CCL250-CONCATENATED STRING EXCEEDS 80 CHARACTERS	Fatal user error. Use of a right arrow (or underline character in ASCII) produced a linked string of more than 80 characters, or a string of 80 characters with a following separator.	Reduce number of linked characters.	CCL
CCL251-DATA COMMAND SPECIFIED CCL FILE-filenam	Fatal user error. File name that user specified on a .DATA command is a CCL working file.	Select another file name.	CCL
CCL252-PROCEDURE CONTAINS NO CONTROL STATEMENTS	Informative message. A procedure should contain at least one control statement.	None.	CCL
CCL261-ENDW WITH MATCHING LABEL FOUND	Nonfatal user error. CCL encountered an ENDW statement in the job control statement record before finding a WHILE statement with a matching label string. A search is initiated for a corresponding WHILE statement. If a WHILE statement with a matching label string is found and WHILE expression is true, normal processing continues with statement following WHILE statement. If WHILE expression is false or if no WHILE statement with a matching label string is found, CCL skips all remaining statements in control statement record.	Check for a WHILE statement and if not present, add. If present, see if WHILE and ENDW label string match and are correct. If they match, place WHILE statement before ENDW statement.	CCL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
CCL262-CONTINUING SEARCH, PRINTING SKIPPED CARDS	Informative message. To assist in isolation of WHILE statement errors such as CCL261, all skipped statements will be printed in user's dayfile until WHILE search is complete.	None.	CCL
CCL263 EXTERNAL ABORT DURING BEGIN.	An external abort during BEGIN processing occurred. Check user's dayfile to find where error occurred.	Correct BEGIN or PROC statement as indicated by dayfile.	CCL
CCL263 EXTERNAL ABORT DURING ENDW.	An external abort occurred during CCL search for the WHILE control statement. The abort is due to a DROP or RERUN command typed by the operator, or an exceeded time limit, mass storage limit, or ECS limit.	Check validation limits. Modify job to fit within limits or ask site to increase the limits.	CCL
CCL270-ERR IN CCL WORK FILES, REVERT TO JOB FILE	Fatal user error. An error was encountered in a file used by CCL because of user manipulation of procedures or data. CCL attempts to return to user's assigned job file. If unsuccessful, the job terminates immediately. If successful, control goes to the EXIT statement and the job terminates. CCL issues this message if a user includes a CLEAR statement or a NEW or OLD statement without the ND parameter in a nested procedure. The CLEAR, NEW, and OLD statements release all temporary files.	Rewrite the job and/or procedures to prevent manipulation of CCL work files. Replace the CLEAR statement with a RETURN statement naming the files to be released. Add an ND parameter to any NEW or OLD statements in the procedure.	CCL
CCL271-REVERT NOT ALLOWED WITHIN JOB FILE	Fatal user error. A REVERT statement cannot appear in control statements of job file.	Remove REVERT statement.	CCL
CCL272-INVALID REVERT PARAMETER - xxx	Fatal user error. Parameter xxx was used on REVERT statement. REVERT statement allows only ABORT parameter or no parameters.	Remove parameter xxx or replace with ABORT.	CCL
CCL300-NON-NUMERIC CHARACTER WITHIN NUMERIC TERM	Fatal user error. An element of an expression which begins with a numeric character cannot contain a nonnumeric character except B or D as a post radix.	Remove any nonnumeric characters other than a post radix B or D.	CCL
CCL302-8/9 DIGIT CONFLICT WITH POST RADIX OF B	Fatal user error. An 8 or 9 is illegal in an octal number.	Remove any 8's or 9's if number is octal or remove post radix B if number is decimal.	CCL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
CCL303-LITERAL NOT TERMINATED	Fatal user error. A literal which begins with a dollar sign was not terminated by a second dollar sign before end of card or line of input was detected.	Place a second dollar sign at end of character string denoted as a literal.	CCL
CCL304-STRCCL - SCATTER BUFFER HEADER INVALID	Fatal system error. This is an internal CCL problem.	Inform site analyst.	CCL
CHANNEL MALFUNCTION, lfn AT addr.	Hardware malfunction.	Inform site analyst.	1MT
CHARGE ABORTED.	Dayfile message indicating that a central site operator action caused the CHARGE operation to abnormally terminate.	Reenter CHARGE statement.	CHARGE
CHARGE FILE BUSY.	Dayfile message indicating that the file which the system uses to validate charge numbers and project numbers is busy.	Reenter CHARGE statement.	CHARGE
CHARGE ILLEGAL AT THIS HOUR.	Dayfile message indicating that the specified project number cannot be used at this time of day.	Retry during the time the project number is valid.	CHARGE
CHARGE NUMBER EXPIRED.	Dayfile and output file message indicating the charge number expiration date has occurred.	None.	CHARGE
CHECK DAYFILE FOR ERRORS.	Informative message indicating that the user should check the dayfile for errors.	Examine error messages in dayfile.	COPY, PFATC, PFCAT, PFCOPY, PFDUMP, PFLOAD, TCOPY
CHECKPOINT nnnn COMPLETE.	Indicates that checkpoint nnnn has completed. Issued if only one checkpoint file is present. For a checkpoint operation, more than two checkpoint files or an illegal combination of checkpoint files was specified.	None.	CHKPT
CHECKPOINT nnnn COMPLETED TO filenam.	Indicates that checkpoint nnnn has been completed to file filenam. Issued if alternate CB checkpoint files are used.	None.	CHKPT
CHECKPOINT FILE ERROR.	During a restart operation, either the checkpoint file specified on the RESTART control statement was empty or RESTART detected a format error during an attempt to read the specified checkpoint file.	Examine checkpoint file to determine error.	CHKPT, RESTART

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
CHECKPOINT NOT FOUND.	The specified checkpoint (nn parameter on RESTART statement) could not be found on the file.	Verify that checkpoint is on file.	RESTART
CIO ERROR.	Updating of resource file returned error status other than end-of-device.	Inform site analyst.	RESEX
CKP REQUEST.	A checkpoint has been initiated.	None.	CHKPT
CM BLOCK OUT OF RANGE.	Data transfer from ECS specified a CM address outside the job field length.	Analyze the job output and dumps to determine the cause.	1AJ
CM NOT VALIDATED.	The number of CM words specified on the job statement exceeds that for which the user is validated.	Rerun job with smaller CM specification or ask site personnel for larger CM validation.	CPM
CM OR EC REQUEST EXCEEDS MAXIMUM.	The user requested a CM field length greater than 377777B or an ECS field length greater than 7777B blocks.	Rerun the job with smaller field length request.	ACCFAM
CM OUT OF RANGE.	The program referenced an address outside the job CM field length.	Analyze job output and dumps to determine the cause.	1AJ
CM PARITY ERROR.	Double data parity error (two data bits failed) between central memory control (CMC) and CM as detected by the single-error correction double-error detection (SECDED) network, or a single parity error when operating in default mode (SECDED network disabled).	Inform customer engineer.	1AJ
CM RANGE EXIT MODE NOT DESELECTABLE.	The user attempted to deselect system checking for CM out of range errors. This cannot be done on a Model 176 system.	Determine if the program can run with CM range error checking. If it cannot, the program must be changed.	CPM
CMC PARITY ERROR.	The CPU sent the central memory control (CMC) data or an address having incorrect parity.	Inform customer engineer.	1AJ
CONFLICTING PARAMETERS.	Input queue type entered more than once.	Reenter with correct parameters.	SUBMIT
CONFLICTING RESOURCE TYPES.	PE, HD, and GE resources cannot be specified concurrently with an NT resource in the same job.	Correct the RESOURC control statement.	RESEX

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
CONNECT REJECT, lfn AT addr.	Unable to connect unit.	Inform site analyst.	1MT
CONTROL CARD ERROR.	An illegal or invalid parameter was specified on the control statement.	Correct control statement and retry.	EDIT, LISTLB
CONTROL STATEMENT LIMIT.	The user has entered too many control statements.	Reduce the number of control statements to a value within the specified limit. Refer to LIMITS command.	TCS
CONTROLLED BACKSPACE ERROR, lfn AT addr.	Controlled backspace operation failed during write error recovery. Position of tape is uncertain.	Inform customer engineer.	1MT
CONVERSION NOT SUPPORTED.	The PC and NC parameters on the FCOPY control statement specify an unsupported conversion.	Refer to the description of the FCOPY statement for valid PC and NC values.	FCOPY
COPY COMPLETE.	Copy termination condition was satisfied before EOI was encountered.	None.	COPY, COPYBR, COPYBF, COPYCF, COPYCR, COPYSBF, COPYX, TCOPY
COPY FL ABOVE USER LIMIT.	Field length for L or F tape copy exceeds user's current maximum.	Use smaller block size on L and F tapes or increase maximum FL.	COPY
COPY INDETERMINATE.	Copy with S, L, or F tape is unpredictable. Only the COPY utility supports these formats.	Retry copy using the COPY control statement.	COPYBF, COPYBR, COPYEI, COPYX
COPYING - type/name	The record with the type and name on the old file was copied to the new file.	None.	COPYL, COPYLM
COPYL COMPLETE.	All records on the old file have been processed.	None.	COPYL, COPYLM
COPYL DID NOT FIND type/name	The record with type and name on the replacement file was not found on the old file. Since the A option was not selected, the record is ignored.	None.	COPYL, COPYLM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
CORE OVERFLOW, JOB ABORTED.	The job field length is too small to hold the tables required to process the program library specified on the UPMOD control statement.	Increase field length and rerun.	UPMOD
CPxx,....	Refer to description of corresponding message beginning with EQ.		
CPM - ARGUMENT ERROR.	Error(s) encountered and job aborted.	Determine error and rerun job step.	CPM
CPM - ILLEGAL PACKNAM.	An illegal pack name has been specified.	Ensure that legal pack name is used.	CPM
CPM - ILLEGAL REQUEST.	A CPM function was issued without the auto recall specified or the job making the request was not of system origin.	Specify auto recall on RA+1 call to CPM or make the job system origin.	CPM
CPM - ILLEGAL USER ACCESS.	The user tried to perform a CPM operation for which he is not validated.	None.	CPM
CPU ERROR EXIT AT addr.	The errors listed after this message occurred at address addr, causing job termination.	Refer to the descriptions of the error messages issued with this message.	1AJ
CRxx,....	Refer to description of corresponding message beginning with EQ.		
CUMULATIVE LIMIT EXCEEDED.	Dayfile and output file message indicating that one of the installation-defined resource usage accumulators for this project exceeded the maximum allowed. The system does not update these accumulators in PROFILa. Each installation must provide this capability if desired.	None.	CHARGE
CUMULATIVE SRU LIMIT EXCEEDED.	Dayfile and output file message indicating that accumulated SRUs have exceeded the maximum allowed.	None.	CHARGE
DATA BASE ERROR.	Dayfile message indicating that the system has detected an error in its validation file.	Contact installation personnel.	CHARGE, MODVAL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
DATA BASE ERROR n - NOTIFY ANALYST.	System error dayfile message indicating that an abnormal situation exists. n is displayed for consideration by the analyst. The internal documentation, obtained by using the DOCMENT control statement, contains an explanation of each error n for use by the analyst. (Refer to section 7 in volume 1 of the NOS Reference Manual for a description of DOCMENT.)	Inform site analyst.	PROFILE
DATA/PERMIT ERRORS, AT addr.	When loading a file from tape, errors were encountered in both data and permit information.	Enter CHANGE statement or macro with CE parameter to allow access to the file. Make the file local and check if data is accurate. Do a CATLIST to see if the permits are accurate.	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
DATA TRANSFER ERROR, AT addr.	An error occurred in a read operation during a file transfer.	Inform site analyst.	PFM
DAYFILE - ARGUMENT ERROR.	Keyword specified is not recognizable or control statement is not properly formatted.	Check keyword and control statement formats.	DAYFILE
DAYFILE - BUFFER TOO SMALL.	The buffer DAYFILE uses to hold the central memory dayfile is not large enough.	Inform the site analyst that either the internal DAYFILE buffer must be made larger or the site must use smaller dayfile buffers for user dayfile message processing.	DAYFILE
DAYFILE - DATA LOST.	A data read error occurred while processing an active file. Processing continues with the next readable message. Lost data is not recoverable.	Inform site analyst.	DAYFILE
DAYFILE - FILE/CM BUFFER BOUNDARY ERROR.	The disk resident portion of the dayfile cannot be joined correctly with the CM buffer portion.	Inform site analyst.	DAYFILE
DAYFILE - FILE TOO LONG.	The active dayfile being processed is longer than the system reservation status indicates.	Inform site analyst.	DAYFILE
DAYFILE - FILE TOO SHORT.	The active dayfile being processed is shorter than the system reservation status indicates.	Inform site analyst.	DAYFILE
DAYFILE - FR INVALID FOR THIS OPTION.	The FR=string parameter is not allowed with the OP=I parameter.	Omit the FR=string parameter and retry the command.	DAYFILE
DAYFILE - ILLEGAL PAGE SIZE FORMAT.	The page size option is nonnumeric.	Retry with a valid option.	DAYFILE
DAYFILE - ILLEGAL PRINT DENSITY FORMAT.	The print density option is nonnumeric. Print density must be 3,4,6, or 8.	Retry with a valid option.	DAYFILE

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
DAYFILE - ILLEGAL PRINT DENSITY FORMAT.	The print density option is non-numeric. Print density must be 3,4,6, or 8.	Retry with a valid option.	DAYFILE
DAYFILE - JOB FIELD DISALLOWED ON USER DAYFILE.	Options specified are not allowed for user dayfile processing.	Retry with a correct OP=op parameter.	DAYFILE
DAYFILE - READ ERROR ON SEARCH FILE.	A read error occurred during the incremental dump and search option processing.	Retry the command.	DAYFILE
DAYFILE - RESERVED FILE NAME.	The file name specified for the L=lfn or I=infile parameter of the DAYFILE command is a reserved name.	Retry using a nonreserved name.	DAYFILE
DAYFILE - TTY INPUT FILE NOT ALOWED.	Dayfile data cannot be entered directly from the keyboard.	Specify dayfile input from other than the terminal (TT device).	DAYFILE
DAYFILE - UNKNOWN *OP* FIELD.	The option specified is not valid.	Retry using a valid option.	DAYFILE
DExx,Ccc,1,sec,ann,Stttt,Aaddr.	An error has been detected on extended core storage. Refer to appendix B of the NOS Operator's Guide for further information.	Inform site analyst.	6DE
DEMAND EXCEEDED.	The user attempted to assign more units than were scheduled on the RESOURC statement.	Increase appropriate parameter value on RESOURC statement.	RESEX
DEMAND FILE ERROR.	Resource execution error was encountered. This error occurred because the demand file (RSXDid) entry does not match the job name.	Inform site analyst.	RESEX
DEMAND VALIDATION ERROR.	The specified number of units exceeds the user's validation limits.	Decrease appropriate parameter value on RESOURC statement.	RESEX
DENSITY CHANGE, TAPE AT nnn.	The tape subsystem detected a change in the data recording density on the tape. Error is due to a malfunctioning tape unit or a bad tape.	Ask customer engineer to examine the tape unit. If tape unit does not require maintenance, discard the tape.	1MT
DEVICE ERROR ON FILE lfn AT addr.	An irrecoverable error occurred on the mass storage device containing the file lfn.	Inform site analyst.	CIO
DEVICE UNAVAILABLE, AT addr.	Access to the permanent file device requested is not possible. User may have attempted to access files on a device not present in the alternate system.	Ask the operator to make the device available.	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
DI....	Refer to the description of corresponding message beginning with EQ.		
DIxx,Ccc,1,sec,ann,Stttt,FNqqqq. or DIxx,Ccc,1,sec,ann,Stttt,Uuu Cyyyy Sttss.	An error has been detected on mass storage device with EST ordinal xx. Refer to appendix B of the NOS Operator's Guide for further information.	Inform site analyst.	7DI
DIRECT ACCESS DEVICE ERROR, AT addr.	The file specified already exists on a device other than the device requested or an illegal device type was specified. The device on which the file resides may not contain direct access files because of one of the following reasons. - The device is not specified as a direct access device in the catalog descriptor 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.	Specify correct device type.	PFM
DIRECTIVE CARD ERROR.	A LIBEDIT directive has incorrect syntax.	Examine the LIBEDIT output to determine cause of error.	LIBEDIT
DIRECTIVE ERRORS.	Dayfile message indicating that one or more input directives were in error. Fatal error.	Examine output file to determine reason for error.	MODIFY, OPLEDIT, LIBTASK, MODVAL, PROFILE, SYSEDIT
DISPLAY DUMP NOT ALLOWED TO TERMINAL.	A time-sharing user has attempted to enter DMD or DMDECS control statements or DMD or DED system requests without assigning file OUTPUT to a mass storage device.	Assign file OUTPUT to mass storage via ASSIGN control statement or macro and retry.	CPMEM
DJ...	Refer to description of the corresponding message beginning with DI or EQ.		
DK...	Refer to description of the corresponding message beginning with DI or EQ.		

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
DL...	Refer to description of corresponding message beginning with DI or EQ.		
DM...	Refer to description of corresponding message beginning with DI or EQ.		
DPxx,Ccc,l,sec,ann,Stttt,FNqqqq. or DPxx,Ccc,l,sec,ann,Stttt,Aaddr,Wwwww DPxx,Ccc,l,Gggg...g. DPxx,Ccc,l,Bbbb...b. or DPxx,Ccc,l,sec,ann,Stttt,Aaddr,Wwwww DPxx,Ccc,l,ddd...d.	An error has been detected on distributive data path (DDP). Refer to appendix B of the NOS Operator's Guide for further information.	Inform site analyst.	6DP
DQ...	Refer to description of corresponding message beginning with DI or EQ.		
DSP - CAN NOT ROUTE JOB INPUT FILE.	The job input file cannot be routed.	Copy job input file to a local file to be routed.	DSP
DSP - COMPLETE BIT ALREADY SET.	The complete bit was not cleared before DSP was called.	Clear complete bit before calling DSP.	DSP
DSP - DEVICE UNAVAILABLE.	DSP attempted to create a file on a device that was turned off or is currently unavailable for access.	Specify different device or contact site operator.	DSP
DSP - FILE NAME ERROR.	An attempt was made to create a file with an invalid file name.	Specify valid file name.	DSP
DSP - FILE NOT ON MASS STORAGE.	An attempt was made to route a file not on mass storage.	Copy file to mass storage before routing.	DSP
DSP - FILE ON REMOVABLE DEVICE.	A file on a removable device cannot be routed.	Copy file to non-removable device before routing.	DSP
DSP - FNT/DEVICE FULL.	There is no space in the FNT or on the device for current use.	Retry route at a later time.	DSP
DSP - FORMS CODE NOT ALPHANUMERIC.	Forms code must consist of two alphanumeric characters.	Specify alphanumeric forms code.	DSP
DSP - I/O SEQUENCE ERROR.	A request was made on a busy file.	Wait until file is not busy.	DSP
DSP - ILLEGAL FILE MODE.	An attempt was made to route an execute only file.	Verify that the file to be routed is not an execute only file.	DSP

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
DSP - ILLEGAL FILE TYPE.	The file being processed is not a print, punch, input, output, or local file type. For example, primary and direct access file types cannot be processed	Ensure that file being processed is of correct type.	DSP
DSP - ILLEGAL ORIGIN TYPE.	DSP cannot route the file to the input queue with the origin type specified by the caller.	Specify valid origin type.	DSP
DSP - ILLEGAL REQUEST.	One of the following. - DSP was not called with recall (does not apply when queue priority is greater than MXPS). - Parameter list address was out of range. - RA+1 call was formatted incorrectly.	Specify auto recall with DSP call or determine why parameter list address is out of range.	DSP
DSP - ILLEGAL USER CARD.	User attempted to route a file with an illegal USER statement to the input queue.	Ensure that valid user number is being used.	DSP
DSP - IMMEDIATE ROUTING - NO FILE.	The specified file for the immediate routing could not be found.	Ensure that file to be routed is available to job for processing.	DSP
DSP - INVALID DISPOSITION CODE.	Specified disposition code is not recognized.	Verify disposition code.	DSP
DSP - INVALID EXTERNAL CHARACTERISTICS.	Caller specified an undefined external characteristic code.	Verify external characteristic code.	DSP
DSP - INVALID TID.	One of the following. - User number and family name parameters were not in CM field length. - TID is greater than or equal to IDLM for batch jobs. - User number specified in parameter block does not compare with user number in control point area.	Verify that TID parameters are valid.	DSP
DSP - LOCAL FILE LIMIT.	User has exceeded his local file validation limits.	Return one or more local files to the system.	DSP
DSP - OUTPUT FILE LIMIT.	Caller has exceeded his output file validation.	If possible, split job into two or more jobs and retry. Otherwise, reduce number of files by copying output to single file and then routing the file.	DSP

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
DSP - ROUTE TO INPUT NOT IMMEDIATE.	Routing a file to the input queue must be immediate.	Change to immediate route.	DSP
DSP - THIS ROUTING NOT ALLOWED.	An attempt was made to change the queue type of a deferred routed file.	Rescind prior routing by using the SC disposition code with the ROUTE control statement or macro. ROUTE the file again with the desired final disposition code.	DSP
DSP - TOO MANY DEFERRED BATCH JOBS.	The user attempted to submit more deferred batch jobs than allowed by his validation limit.	Wait for jobs to complete or request a larger validation limit from site personnel.	DSP
****DUPLICATE CHARGE NUMBER.	Output file message indicating that an existing charge number was referenced on a create run.	Rerun using correct charge number, if required.	PROFILE
DUPLICATE COMMON FILE NAME.	A file of the same name as that specified in a COMMON request already exists.	Use different name in request.	LFM
DUPLICATE FILE NAME.	The file specified already exists in the system.	Use different name in request.	LFM
****DUPLICATE PROJECT NUMBER.	Output file message indicating that an existing project number was referenced on a create run.	Rerun using correct project number, if required.	PROFILE
DUPLICATE USER NUMBER.	Output file message indicating that the user number encountered on a create run is a duplicate of a user number previously entered. The first entry is used.	Rerun the corrected job or correct the new validation file, if necessary.	MODVAL
DUPLICATED LINES.	Duplicate lines being dumped during a DMP operation were suppressed.	None.	CPMEM
EC NOT VALIDATED.	The number of ECS blocks specified on the job statement exceeds that for which the user is validated.	User should check his validation with the LIMITS statement.	CPM
ECS BLOCK OUT OF RANGE.	Data transfer between CM and ECS specified an ECS address outside the job field length.	Analyze the job output and dumps to determine the cause of the error.	1AJ
ECS ERROR ON ROLLOUT.	An irrecoverable ECS parity error occurred while the system was writing the job's ECS field length to a rollout file.	Inform the site analyst of the parity error. Rerun the job after ECS has been repaired.	1RI

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ECS FLAG REGISTER PARITY.	Parity error detected on ECS flag register operation.	Inform customer engineer.	1AJ
ECS OUT OF RANGE.	Job referenced ECS address outside job field length.	Analyze the job output and dumps to determine the cause of the error.	1AJ
ECS READ ERROR.	An unrecoverable ECS read error occurred.	None.	CPMEM
EMPTY CATALOG.	No entries are present in the catalog.	None.	CATLIST, NDA
EMPTY LINES.	Blank lines were encountered.	Delete or replace the empty lines.	NOTE

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
EMPTY MSORT INPUT FILE.	Dayfile message indicating that the file specified on the SORT control statement contains no data.	Correct and rerun.	MSORT
EMPTY SORT INPUT FILE.	Dayfile message indicating that the file specified on the SORT control statement contains no data.	Correct and rerun.	SORT
END OF INFORMATION ENCOUNTERED.	An end-of-information mark was encountered on the input file.	None.	COPYCF, COPYCR, COPYSBF
END OF SET ILLEGAL REQUEST, lfn AT addr.	A multifile set tape is positioned at the end of set after searching for a file set that was not found. Until the tape is returned or repositioned within the multifile set, all other tape operations on this tape are illegal.	Return the tape via a RETURN, UNLOAD, or EVICT statement, or reposition the tape with the LABEL statement.	1MT
END OF TAPE, lfn AT addr.	The end-of-tape was encountered.	Ensure that correct file manipulation operation is specified.	1MT
**** ENDING SUPPORT OF *OUT* RA+1 REQUEST.	Warning message indicating that this request will not be available in future versions of NOS.	Convert the program to use DSP to queue files.	OUT
ENQUIRY COMPLETE.	Message issued when ENQUIRE control statement processing is completed.	None.	ENQUIRE
ENTRY POINT NOT FOUND.	The specified entry point could not be found.	Verify that entry point is valid.	1AJ
EOF ENCOUNTERED.	End-of-file was encountered before copy termination condition was satisfied.	None.	COPYX
EOF ENCOUNTERED BEFORE TERMINATION.	An end-of-file was encountered on a CONVERT input file before the specified record count was reached.	Ensure accuracy of input file.	CONVERT
EOI CHANGED BY RECOVERY, AT addr.	The file was truncated during deadstart recovery.	Use CHANGE statement or macro with CE parameter to allow access to the file. Make the file local and list it to determine how much of the file was lost.	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
EOI ENCOUNTERED.	End-of-information was encountered on the input file.	If the message was not expected, check if the copy was performed as desired. The source file may not have been positioned correctly before the copy.	COPY, COPYBF, COPYBR, COPYCF, COPYCR, COPYEI, COPYSBF, COPYX, TCOPY
EOI ENCOUNTERED BEFORE TERMINATION.	An end-of-information was encountered on a CONVERT input file before the specified record count was reached.	Ensure accuracy of input file.	CONVERT
EQxx,CHcc Fffff FUNCTION TIMEOUT.	No response (inactive) was received after a function code was issued to the specified local batch equipment (converter and equipment status unavailable). EQ One of the following equipment types. CP 415 card punch CR 405 card reader LP Any line printer LR 580-12 line printer LS 580-16 line printer LT 580-20 line printer xx EST ordinal of local batch equipment cc Channel number ffff Function code	Inform customer engineer.	QAP, 1CD
EQxx,CHcc Fffff REJ Paaaa,Cbbbb,Emmmm.	Function reject or transmission parity error was detected on the specified local batch equipment. EQ One of the following equipment types. CP 415 card punch CR 405 card reader LP Any line printer LR 580-12 line printer LS 580-16 line printer LT 580-20 line printer xx EST ordinal of local batch equipment cc Channel number ffff Function code aaaa Driver (1CD) address bbbb Converter status mmmm Equipment status	Inform customer engineer.	QAP, 1CD
EQxx,CHcc, PRINT ERROR LIMIT EXCEEDED.	Maximum number of consecutive print errors was detected on line printer xx. EQ One of the following equipment types. LP Any line printer LR 580-12 line printer	Inform customer engineer.	1CD, QAP

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	LS 580-16 line printer LT 580-20 line printer xx EST ordinal of line printer cc Channel number		
EQxx,CHcc RESERVED.	The specified local batch equipment is reserved and cannot be connected on channel cc. EQ One of the following equipment types. CP 415 card punch CR 405 card reader LP Any line printer LR 580-12 line printer LS 580-16 line printer LT 580-20 line printer xx EST ordinal of local batch equipment cc Channel number	Inform customer engineer.	IIO
EQxx,DNdn, DIRECT ACCESS FILE ERROR, AT addr.	The system sector data for the file does not match the catalog data. Error log and dayfile message. EQ One of the following equipment types. DI 844-21 disk (half track) DJ 844-41/44 disk (half track) DK 844-21 disk (full track) DL 844-41/44 disk (full track) DM 885 disk (half track) DQ 885 disk (full track) xx EST ordinal of device dn Device number	Inform site analyst.	PFM
EQxx,DNdn, FILE LENGTH ERROR, AT addr.	The length of a file does not equal the length specified in the catalog. EQ Equipment type as defined in EQxx,DNdn, DIRECT ACCESS FILE ERROR, AT addr. message xx EST ordinal of device dn Device number The cause depends on the type of command or macro issued. GET A local file is created with length being the actual length retrieved. SAVE If file length is longer than TRT specification, file is truncated. REPLACE Same as for SAVE.	Inform site analyst.	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
EQxx,DNdn, MASS STORAGE ERROR AT addr.	An error was encountered in reading a portion of the permanent file catalog or permit information. Error log and dayfile message. EQ Equipment type as defined in the EQxx,DNdn, DIRECT ACCESS FILE ERROR, AT addr. message xx EST ordinal of device dn Device number	Inform site analyst.	PFM
EQxx,DNdn, RANDOM INDEX ERROR, AT addr.	The random disk address of the permit sector is in error. Error log and dayfile message. EQ Equipment type as defined in the EQxx,DNdn, DIRECT ACCESS FILE ERROR, AT addr. message xx EST ordinal of device dn Device number	Inform site analyst.	PFM
EQxx,DNdn, REPLACE ERROR, AT addr.	The same file was found twice during a catalog search. This error can occur for APPEND or REPLACE commands or macros after a file is found and purged and the catalog search is continued. Error log and dayfile message. EQ Equipment type as defined in the EQxx,DNdn, DIRECT ACCESS FILE ERROR, AT addr. message xx EST ordinal of device dn Device number	Retry job step.	PFM
EQxx,DNdn, SYSTEM SECTOR ERROR, AT addr.	The system sector of an indirect access permanent file contains an error. Error log and dayfile message. EQ Equipment type as defined in EQxx, DNdn, DIRECT ACCESS FILE ERROR, AT addr. message description xx EST ordinal of device dn Device number	Inform site analyst.	PFM
EQxx,DNdn, TRACK LIMIT, AT addr.	No allocatable tracks remain on equipment xx. Error log and dayfile message. EQ Equipment type as defined in the EQxx,DNdn, DIRECT ACCESS FILE ERROR, AT addr. message xx EST ordinal of device dn Device number	Inform site analyst.	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
EQxx,FM=family,PF=filenam,UI=userin.	Additional line is written only in error log after one of the following messages. EQxx,DNdn,DIRECT ACCESS FILE ERROR, AT addr. EQxx,DNdn,FILE LENGTH ERROR, AT addr. EQxx,DNdn,MASS STORAGE ERROR, AT addr. EQxx,DNdn,RANDOM INDEX ERROR, AT addr. EQxx,DNdn,REPLACE ERROR, AT addr. EQxx,Dndn,SYSTEM SECTOR ERROR, AT addr. EQxx,DNdn,TRACK LIMIT, AT addr. EQ Equipment type as defined in the EQxx,DNdn, DIRECT ACCESS FILE ERROR, AT addr. message xx EST ordinal of device family Family name filenam Permanent file name userin User index	None.	PFM
EQxx,nnnn PRINT ERRORS.	Print errors detected on line printer xx. EQ One of the following equipment types. LP Any line printer LR 580-12 line printer LS 580-16 line printer LT 580-20 line printer xx EST ordinal of line printer nnnn Octal number of print errors	Inform customer engineer.	1CD, QAP
EQUIPMENT NOT AVAILABLE.	Tape assignment error was encountered; requested equipment is either in use or is not defined in the system.	Ensure accuracy of macro or control statement and/or retry at a later time.	LFM, RESEX
ERASE LIMIT, lfn AT addr.	The system made 20 erasures (10 feet of tape) without being able to successfully write the tape.	Clean tape or use different tape.	1MT
ERROR AT LINE xxx.	Errors have occurred while resequencing a BASIC program. The line containing the error is specified by xxx.	Examine line xxx of program to determine cause of error.	RESEQ
ERROR CODE ec, lfn AT addr.	1MT error code ec has occurred but no specific message is issued. This would normally not occur unless the job was dropped by the operator.	Consult site personnel.	1MT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ERROR - FILES(S) NOT PROCESSED.	One or more files were not checkpointed because CHKPT detected address errors.	Determine and correct address errors and retry.	CHKPT
ERROR FLAG TERMINATION, lfn AT addr.	The job was aborted with a tape operation in progress. The operation/request is not complete. For example, the operator could kill the job while tape error recovery is in progress.	None.	IMT
ERROR FLAG TERMINATION, FILE lfn AT addr.	The job was aborted with an input/output request in progress. The operation/request is not complete.	None.	CIO
ERROR IN ARGUMENT.	One of the following. - The pfn is blank or fn = lfn. - The user specified no arguments or a blank argument. - The user specified too many files. - The user entered an illegal parameter.	Reenter the command or control statement with correct parameters.	PFILES
ERROR IN ARGUMENTS.	One or more of the following control statement errors were detected. - More than one date was entered. - No options were selected. - The parameter was illegal or could not be recognized. - The TM option was selected but no data was specified. - Both the device number parameter and the pack name or auxiliary device parameter were selected; auxiliary devices do not have device numbers.	Correct control statement and retry.	BLANK, PURGALL, ITEMIZE
ERROR IN BACKUP REQUIREMENT.	The option specified on the BR parameter on the DEFINE, CHANGE, or SAVE statement is incorrect.	Check the BR parameter description and correct the statement.	PFILES
ERROR IN CONTROL ARGUMENTS.	Either the number of arguments was illegal, an illegal argument was specified, or a numeric parameter was nonnumeric.	Check the description of the control statement.	CONTROL
ERROR IN DATE.	The format of the date (ad, md, or cd) parameter in a PURGALL request was incorrect.	Check the format of the PURGALL statement in section 8.	PURGALL
ERROR IN DEVICE NUMBER.	The file residency as specified by the device number parameter was illegal.	Correct the device number parameter.	PURGALL

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ERROR IN FCOPY ARGUMENTS.	The format of the FCOPY control statement is in error.	Refer to the description of the FCOPY control statement.	FCOPY
ERROR IN FILE ARGUMENTS.	The system did not recognize one or more parameters.	Compare the parameters specified with the command or control statement description.	MFILES
ERROR IN FILE CATEGORY.	The user specified an illegal file category.	Refer to description of file categories for valid entry.	PFILES, PURGALL
ERROR IN FILE DATA, AT addr.	When loading a file from tape, an error was encountered in the data.	Enter CHANGE statement or macro with CE parameter to allow access to the file. Make the file local and ensure the accuracy of the data.	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ERROR IN FILE TYPE.	The user specified an illegal file type.	Check the PURGALL statement description and correct the statement.	PURGALL
ERROR IN LIMITS ARGUMENT.	Dayfile message indicating that parameters were included on the LIMITS statement.	Enter LIMITS. without additional parameters.	MODVAL
ERROR IN PASSWOR ARGUMENTS.	PASSWOR control statement or command parameters are incorrect.	Correct control statement or command and reenter.	PASSWOR, MODVAL
ERROR IN PERMIT DATA, AT addr.	When loading a file from tape, an error was encountered in the permit entries.	Enter CHANGE statement or macro with CE parameter to allow access to the file. Do a CATLIST to see if the permits are accurate.	PFM
ERROR IN PREFERRED RESIDENCE.	The option specified on the PR parameter on the DEFINE or CHANGE statement is incorrect.	Check the PR parameter description and correct the statement.	PFILES
ERROR IN PROFILE ARGUMENTS.	Dayfile message indicating there was an error on the PROFILE control statement.	Correct the control statement and rerun.	PROFILE
ERROR IN ROUTE FUNCTION, LFN=filenam.	Informative message issued to the system dayfile stating an error occurred while routing filenam.	Examine the job's dayfile for a more specific error message.	DSP
ERROR IN SUBSYSTEM.	The option specified on the SS parameter on the CHANGE or SAVE statement is incorrect.	Check the SS parameter description and correct the statement.	PFILES
ERROR IN SYSTEM TEXT.	The system text referenced by the KRONREF control statement did not contain symbol definitions.	Ensure that the system text specified by the G or S parameter contains symbol definitions.	KRONREF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ERROR IN TIME.	The format of the time parameter in a PURGALL request was incorrect.	Check the format of the PURGALL statement in section 8.	PURGALL
ERROR IN 5TH COPYL PARAMETER	Fatal error; only A, R, E, and T are recognized parameters.	Correct processing option.	COPYL, COPYLM
ERROR LIMIT EXCEEDED.	Number of parity and block-too-large errors exceed the error limit.	If dayfile shows block-too-large errors have occurred and tape is S, L, or F format, increase block size and retry; otherwise, tape is probably assigned in the wrong format. If parity errors have occurred, the tape is bad and the data on it cannot be correctly recovered.	COPY, TCOPY
EXCHANGE PACKAGE/MEMORY DUMP ON FILE ZZZDUMP.	The exchange package and memory dump is written on local file ZZZDUMP because the job is of time-sharing origin and file OUTPUT is assigned to a terminal.	To examine the exchange package and dump, list file ZZZDUMP.	CPMEM
EXU ARG. ERROR.	The address of the arguments is out of range or the program name is zero.	Correct error and retry.	EXU
FCOPY COMPLETE.	The FCOPY conversion is finished.	None.	FCOPY
FILE BOI/EOI/UI MISMATCH, lfn AT addr.	Data in the system sector for file lfn does not match information from the EOI sector and/or catalog information.	Inform site analyst.	PFM
FILE BUSY.	The file specified in the request cannot be interlocked.	Wait until file is not busy.	LFM
FILE EMPTY.	The Update program library contained no data.	None.	UPMOD
FILE EMPTY.	The file specified was empty.	Verify that the file contains data and retry.	LFM, SFM
FILE ERROR filenam.	An illegal address was detected on file filenam.	Correct error and retry.	CHKPT, RESTART
FILE NAME CONFLICT.	Either the user tried to process two files having the same name or he specified a reserved file name.	Correct the control statement so all files have a unique name.	CONVERT, FCOPY, TCOPY, COPY, VERIFY,

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
			LIBGEN, UPMOD
FILE NAME CONFLICT.	For the LO72 control statement, the S and L file names are the same. For the ITEMIZE control statement, the output and binary file names are the same.	Rename either output file or file to be itemized.	LO72, ITEMIZE
FILE NAME ERROR.	File name contains illegal characters or contains more than seven characters.	Ensure that legal file name is specified.	LFM, LIBGEN, LISTLB
FILE NAME ERROR, AT addr.	File name contains illegal characters.	Verify that file name contains only valid characters.	PFM
FILE NOT FOUND - filenam.	Warning message that informs the user that the designated file did not exist prior to the copy or verify.	If file should have existed, reaccess the file and retry copy.	COPY, COPYBR, COPYBF, COPYEI, COPYX, TCOPY, VERIFY
FILE NOT FOUND.	Requested file could not be found.	Verify that file exists and retry.	LFM, SFM, QFM, ENQUIRE, STIMULA
FILE NOT ON MASS STORAGE.	The specified file does not reside on mass storage.	Copy file to mass storage and retry.	1AJ
FILE STRUCTURES NOT COMPATIBLE.	Verification results cannot be guaranteed correct when the logical structures of the files being verified are not compatible.	None.	VERIFY
FILE TOO LONG.	File cannot be sorted in memory. This message can also indicate that the files were out of order.	Either rerun with the files in the correct order or submit a sort job.	MSORT
FILE TOO LONG, AT addr.	The local file specified for a SAVE, REPLACE, or APPEND command exceeds the length allowed or the direct access file specified for an ATTACH operation in WRITE, MODIFY, or APPEND mode exceeds the direct access file length limit for which the user is validated.	Reduce length of file or save as a direct access file.	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
FILE TOO LONG, FILE lfn AT addr.	The length of the direct access permanent file currently being written exceeds the direct access file length limit for which the user is validated.	Reduce the length of the file or ask site personnel for a larger direct access file length limit.	CIO
FILENAME CONFLICT.	The first two parameters of the GTR control statement are identical.	Specify different file names for the first two parameters on the GTR statement.	GTR
FL BEYOND MFL (ECS).	ECS field length requirements for the job step exceed the ECS field length allowed.	Increase job step ECS field length.	IMA
FL REQUEST BEYOND MFL (CM).	CM field length requirements for the job step exceed the CM field length allowed.	Increase job step CM field length.	IMA, TCS
FL TOO SHORT FOR PROGRAM.	The user's field length is too short for the program.	Rerun the job with larger field length specification.	TCS, IAJ
FLE TOO SHORT FOR LOAD.	An attempt was made to load ECS data beyond the user's ECS field length.	None.	IAJ, LDR
FM NOT LEGAL FAMILY.	Dayfile message indicating that an illegal family name was specified with the FM parameter.	Correct the FM parameter and rerun.	PROFILE, MODVAL
FORMAT ERROR.	One of the following. - The GTR control statement format was incorrect. - An illegal library type was specified. - A record name longer than seven characters was specified.	Refer to the description of the GTR control statement.	GTR
FORMAT ERROR ON CONTROL CARD.	An error was detected in the format of the control statement.	Check the description of control statement format.	TCS
FORMAT REQUIRES UNLABELED TAPE.	The format specified (F) is valid only for unlabeled tapes.	The tape must be assigned as an unlabeled tape.	RESEX
FR INVALID FOR THIS OPTION.	The user entered a FR=string parameter with the OP=I parameter (incremental dump).	Check the DAYFILE statement description and correct the statement.	DAYFILE
FR NOT FOUND.	The specified string was not in the dayfile. A time-sharing user will get a full dayfile dump. A dayfile dump is not produced for a batch user.	Retry with a corrected FR=string.	DAYFILE

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
FUNCTION REJECT, lfn AT addr.	Function was rejected (possible hardware problem).	Inform site analyst.	1MT
FWA .GE. LWA+1.	The first word address parameter was greater than the last word address parameter on DMP, DMD, DMPECS, DMDECS, LOC, or PBC control statements or DMP, DMD, DED, or DEP system requests.	Correct error and retry.	CPMEM
FWA/LWA .GE. FL.	Either first word address parameter of LOC or the last word address of LOC or PBC was greater than or equal to the user's field length.	Reduce FWA and/or LWA and retry.	CPMEM
GTR ERRORS.	GTR detected an error with the call.	Check the description of the GTR statement.	GTR
H VALUE INVALID.	The H parameter was zero or greater than the buffer length.	None.	LO72
HTIME xxxxxxxxxxx.xxx KCYCLES.	Dayfile message giving the CYBER 170 Model 176 CPU clock cycle count for the job. The count is in kilocycle units.	None.	1AJ
HTIME NOT AVAILABLE.	The HTIME control statement or macro is valid only on a CYBER 170 Model 176.	None.	1AJ
I/O ON EXECUTE-ONLY FILE lfn AT addr.	The user attempted to read, write, or position an execute-only file. The only operations that may be performed on an execute-only file are EVICT, RETURN, REWIND, and UNLOAD.	Assign file in proper mode to allow the desired operation.	CIO
I/O SEQUENCE ERROR.	Action was requested on a busy file.	Wait until file is not busy and retry.	CFM, LFM
I/O SEQUENCE ERROR, AT addr.	A request was attempted on a local file that is currently active. This error can occur, r for example, if the user creates two FETs fo the same file and issues a second request before the first is completed.	Wait until file is inactive.	PFM, LFM
I/O SEQUENCE ERROR ON FILE lfn AT addr.	The user attempted to perform more than one concurrent function on a single file.	Wait until each function is complete before attempting another.	CIO
ILLEGAL CHANGE IN FILE/ORIGIN TYPE.	LFM attempted to change the file/origin type of a deferred routed output file.	Rescind prior routing (DC=SC option).	LFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ILLEGAL CHARACTER NUMBER.	In a copy request, one of the following was detected. - Last character position was less than first character position. - Last character position was greater than 150. - Either first character position or last character position was unrecognizable.	Ensure accuracy of control statement parameters.	COPYCF, COPYCR
ILLEGAL CHARGE.	Dayfile and output file message indicating one of the following. - The charge or project number does not exist. - The project number is not available to a user with this user number. - The charge or project number exists but is inactive.	Check to see that charge and project numbers are correct and reenter.	CHARGE
ILLEGAL CODE, FILE xxxx, REC yyyy.	In file xxxx, record yyyy of lfn1 (as named in the FCOPY control statement) a code exists which does not have a corresponding code in the character code set of lfn2. FCOPY converts the code to a space and continues converting lfn1.	List lfn1 using the TDUMP control statement to determine the code converted to a space. Refer to appendix A for listings of the character code sets.	FCOPY
ILLEGAL COMBINATION OF QN/SI.	SI was specified and QN was not. If QN is specified, then SI must also be specified.	Specify SI and rerun the job.	LISTLB
ILLEGAL CONTROL CARD.	One of the following has occurred. - The control statement could not be identified. - The USER control statement does not have a user number specified. - An invalid parameter was specified or no terminator was detected. - The user included too many parameters on the program call statement or the program was not present. - The user submitted a control statement that he was not validated to use (for example, the use of PASSWOR by user not validated to change password). - The user submitted a control statement that is illegal for a particular job type or file type (for example, the use of a FAMILY statement in a nonsystem origin job).	Ensure accuracy and/or suitability of control statement.	TCS, CHARGE, CONFIG, MODVAL, RESEX, EXU, PASSWOR, 026

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ILLEGAL COPY.	File and/or conversion types do not meet copy requirements.	Correct error and retry.	COPY, TCOPY
ILLEGAL COUNT.	The copy file count is nonnumeric or specified as zero.	Correct control statement and retry.	COPYCF, COPYCR, COPYSBF
ILLEGAL DEVICE REQUEST, AT addr.	The device type (r parameter) specified on a request for an auxiliary device cannot be recognized or does not exist in the system. If the auxiliary device specified by the pn parameter is not the same type as the system default, the r parameter must be included; if not, the message is issued.	Examine auxiliary device request and ensure its accuracy.	PFM
ILLEGAL DISPOSE CODE.	The queue type specified on a DISPOSE control statement was unrecognizable.	Consult description of DISPOSE control statement for valid queue types.	FILES
ILLEGAL EQUIPMENT.	Equipment specified does not exist or is not allowed (for example, a TT device is requested from other than terminal origin, or a tape is being requested with the REQUEST macro).	Ensure file resides on a legal equipment type.	LFM, RESEX
ILLEGAL ERROR EXIT ADDRESS.	Error exit address is beyond user's current field length.	Informative.	1AJ
ILLEGAL EXTENSION OF lfn AT addr.	The user attempted to lengthen a file that could not be extended.	Verify that valid file is being extended.	CIO
ILLEGAL EXTERNAL CALL.	RESEX did not recognize external call.	Inform site analyst.	RESEX
ILLEGAL FILE COUNT.	The file count was not numeric.	Correct the N=n parameter.	ITEMIZE
ILLEGAL FILE MODE.	The user tried to dispose or unlock a file which was in execute-only mode or tried to change its file type to library (LIFT).	None.	LFM
ILLEGAL FILE NAME lfn AT addr.	The file name does not conform to established rules.	Use valid file name.	CIO
ILLEGAL FILE TYPE.	The specified file is of a type not allowed in the requested operation. Possible causes include attempts to - change a nonlocal file to file type library - designate a direct access file as the	Verify that file type is valid.	LFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	primary file - ROUTE or DISPOSE the primary file		
ILLEGAL HOLL. CODE, RECxxx CDyyyy.	Statement yyyy (octal count) in record xxx was found to contain an illegal Hollerith code. The job was terminated without EXIT processing.	Correct statement in error.	1AJ
ILLEGAL I/O REQUEST ON FILE lfn AT addr.	CIO could not recognize the specified function code, or the code was not valid for the type of device to which the file was assigned. The system provides a dump of the FET on file OUTPUT.	Verify CIO function code being used.	CIO
ILLEGAL ID CODE.	One of the following. - An identification code specified on the SETID control statement or macro is not a valid device identification code as defined in the installation EST. - An identification code not in the range 0-67B or 77B is present on the LDI control statement.	Reissue the request with the correct identification code.	LFM, LDI
ILLEGAL INPUT FILE.	An attempt was made to pack a file that is assigned to a time-sharing terminal. For example, file INPUT for time-sharing origin jobs cannot be packed.	Verify that file being packed is not assigned to terminal (TT).	PACK
ILLEGAL INSTRUCTION.	The CPU attempted to execute an illegal or nonavailable instruction.	Analyze job output and dumps to determine the cause of the error.	1AJ
ILLEGAL JOB ORIGIN TYPE FOR TRMDEF.	The TRMDEF control statement was entered from a job that was not of time-sharing origin.	Retry from a time-sharing origin job or remove the TRMDEF statement.	TRMDEF
ILLEGAL JOB/USER CARDS.	A job was submitted with an invalid job or user card.	Correct job and rerun.	QFM
ILLEGAL LABEL TYPE, lfn AT addr.	Illegal label type. Only legal label types are ANSI labeled and nonstandard labeled.	Use correct label type.	1MT
ILLEGAL LOAD ADDRESS.	The load address is less than 2.	Specify larger load address and retry.	1AJ
ILLEGAL MODIFICATION OF lfn AT addr.	Either the user has attempted to shorten a modify-only file or the file cannot be modified at all.	Determine whether file can be modified.	CIO

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ILLEGAL OPTION.	The option specified is not defined for ENQUIRE.	Check parameters on control statement and retry.	ENQUIRE
ILLEGAL PARAMETER.	One of the following. - Parameter value is outside legal bounds. - The user specified a parameter that cannot be included on the command or control statement. - Command/control statement is invalid.	Ensure accuracy of command/control statement.	IAFEX, LISTLB, CATLIST, TELEX
ILLEGAL PASSWORD.	One of the following. - The password entered is greater than seven characters or contains an invalid character. - In the PASSWOR command either an incorrect old password was specified or the new password was unacceptable. - In the MODVAL control statement (for a create or update run) the password for a new user contained fewer characters than the minimum length required by the site. If entered from a K display, the line of input is ignored; otherwise, that particular user number is disregarded.	Correct error and retry.	PASSWOR, MODVAL, PFILES
ILLEGAL QUEUE SPECIFIED.	Queue type specified is illegal.	Retry with valid queue type.	SUBMIT
ILLEGAL REQUEST.	No parameters were specified on a DMPECS or DMDECS control statement.	Retry job with corrected control statement.	CPMEM
ILLEGAL RESOURCE COUNT.	Total resource demand exceeds maximum allowed, as defined by installation parameter MAXD (defined in RESEX).	Reduce RESOURC demands.	RESEX
ILLEGAL SORT PARAMETER.	An illegal parameter has been specified on the SORT control statement.	Consult description of SORT control statement for valid parameters.	SORT
ILLEGAL TERMINAL REQUEST.	A command intended for time-sharing origin jobs only has been used in a nontime-sharing origin job.	None.	IAFEX, TELEX
ILLEGAL TERMINATION CONDITION.	The specified termination record count or record type was not recognized.	Refer to the COPYX statement description, correct the error, and retry.	COPYX

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ILLEGAL USER ACCESS.	The user tried to perform an operation for which he is not validated. Possible causes include attempts to - run a system origin job from nonsystem origin - access a restricted subsystem without proper validation - enter an invalid SRU value - use the V carriage control character without validation	Ensure accuracy of control statement or determine proper validation requirements via LIMITS statement.	LFM, MSI, NETVAL, QFSP, RESEX, 1MA
ILLEGAL USER ACCESS, AT addr.	The user is not validated to create direct access or indirect access files or to access auxiliary devices.	Contact site personnel concerning validations.	PFM
ILLEGAL USER ACCESS - CONTACT SITE OPR.	The security count for the user number specified has been decremented to zero. Therefore, the user is denied all access to the operating system until the operator resets the security count.	Contact site personnel to reestablish access.	CPM, NETVAL, 1AJ, 1LS
ILLEGAL USER CARD.	The user attempted to submit a file which does not have a USER statement as the second statement, has an incorrectly formatted USER/ACCOUNT statement, or has an invalid family name, user name, or password.	Correct error and rerun.	QFM
ILLEGAL USER CARD.	The user number or password could not be validated, or a secondary user statement was encountered while secondary user statements were disabled. Dayfile message.	Verify that user number and password are valid. If secondary user statements are disabled, ensure that no secondary user statements are present.	CPM, ACCFAM
IMPROPER ACCESSIBILITY.	The user did not specify the correct file accessibility on the LABEL statement or macro, or volume accessibility was set and a nonsystem origin user attempted to assign the tape as unlabeled.	Ensure accuracy of request.	RESEX

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
IMPROPER VALIDATION.	A validation program (one containing a VAL= entry point, such as that used for CHARGE and USER) is required before continuing.	Verify that USER statement precedes rest of job (followed by CHARGE, if required) and is the first statement after the job card.	TCS
INITIAL CONTROL STATEMENT LIMIT.	The number of control statements processed by the job exceeded the limit for which the user is validated. At this point, an additional eight statements are allowed for error processing.	Split the job into two or more jobs, reduce the number of control statements in the job, or request a larger control statement limit from site personnel.	1AJ
INITIAL MESSAGE LIMIT.	The number of messages the job entered in the dayfile exceeded user's limit. Eight additional messages are allowed for error processing.	Restructure job to reduce dayfile messages.	1MA
INPUT FILE ERROR.	An error on the input file was encountered during the unpack operation.	None.	LO72
INPUT FILE IN NORERUN STATUS.	Informative message.	None.	QFM
INPUT FILE IN RERUN STATUS.	Informative message.	None.	QFM
INQUIRY COMPLETE.	Dayfile message indicating that the inquiry is completed.	None.	MODVAL
INSUFFICIENT RESOURCES ON SYSTEM.	Resource demand exceeds number of units physically available on the system.	Reduce resource demand.	RESEX
INSUFFICIENT STORAGE FOR LIBRARY GENERATION.	Additional memory is required for the LIBGEN control statement.	Increase field length and retry.	LIBGEN
INVALID LINE LENGTH.	Either of the following out of bounds conditions exists with respect to the Ix, Nx, Ox, and H parameters of the LO72 control statement. - (Ox+Nx).GT.H - (Ix+Nx).GT.(buffer length) where 1 .LE. x .LE. b	None.	LO72
INVALID NOISE SIZE ON filenam.	Noise size on S, L, or F format tape does not meet copy requirements.	For copy, ensure that noise size on input file is greater than or equal to that of the output file. For TCOPY, correct noise size to meet requirements and	COPY, TCOPY

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
		retry (noise size of 8 for 7-track or 6 for 9-track tapes is required for X and SI format conversions).	
INVALID PARAMETER.	The S or L parameter was entered as zero on the L072 control statement.	None.	L072
INVALID SPACING CODE.	The SC=xx parameter on the ROUTE statement specified a spacing code greater than 77B.	Correct the spacing code specified on the ROUTE statement.	ROUTE
ITEMIZE COMPLETE.	Specified processing is finished.	None	ITEMIZE
ITEMIZING xxxx	Record xxxx is being processed.	None.	ITEMIZE
IX OR OX NOT DEFINED.	The Ix or Ox parameter was not spcified in conjunction with the Nx parameter on a L072 control statement.	Check the description of the L072 control statement in section 7.	L072
JOB CARD ERROR (job statement)	An error was encountered on the job statement in the routed or submitted file. The first 20 characters of the job statement are displayed.	Correct job statement and rerun.	DSP, QFM
JOB CARD ERROR.	The job statement on the file being submitted is in error.	Compare the job statement in error with the job statement description in section 5. Also check your user validation limits.	IAJ
JOB IN NORERUN STATE ON RECOVERY.	Identifies a job recovered on level 0 deadstart that was aborted because it was in a no-rerun mode (due to NORERUN control statement or macro).	Refer to the NORERUN control statement or macro.	IAJ
JOB REPRIEVED.	The job has been successfully reprieved.	None.	SFP
JOB STEP EXCEEDS ACCOUNT BLOCK.	The user tried to set his job step limit to a value greater than his account block limit or tried to set his account block limit to a value less than his job step limit.	Check values on SETJSL and SETASL statements.	CPM
JOB STEP LIMIT.	The monitor detected the expiration of the job step SRU limit.	Reset job step limit with SETJSL control statement or macro and retry. If job step SRU limit is set at maximum, request increased SRU validation.	IAJ

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
JOBNAME IS jobname	An LDI statement entered a file in the input queue. The entered job has the system job name of jobname.	None.	LDI
LABEL CONTENT ERROR, lfn AT addr.	A block read was the correct size for a label but one or more required fields (such as the label name) were incorrect.	Use LISTLB control statement to obtain label data.	1MT
LABEL MISSING, lfn AT addr.	During a read operation, a required label was missing.	Ensure that tape has label.	1MT
LABEL NOT EXPIRED.	The user attempted to write on a tape with an unexpired label.	If current contents of tape can be sacrificed, have operator blank label tape. Otherwise, wait until label has expired.	1MT
LABEL PARAMETER ERROR ON OPEN, lfn AT addr.	Label fields did not match on open request. An additional message FIELD BEGINNING AT addr NO COMPARE. specifying the decimal character position in HDR1 of the first field that did not compare correctly is also issued.	Use LISTLB control statement to obtain label data.	1MT
LDR ERROR.	Issued after one of the following errors. OVERLAY NOT FOUND IN LIBRARY. ARGUMENT ERROR.	Correct error and retry.	LDR
LFM ARG. ERROR.	LFM detected an error in the request.	Ensure that valid LFM request is being made.	LFM
LFM ILLEGAL REQUEST.	One of the following. - The LFM function detected was not recognized as a legal function. - An LFM function was issued without the auto recall bit set.	Verify that valid LFM request is being used.	LFM
LIBGEN ARGUMENT ERROR.	An invalid parameter was used on the LIBGEN control statement.	Check the format of the LIBGEN control statement.	LIBGEN
LIBGEN FILE NAME CONFLICT.	The LIBGEN statement named the same file as the input file and as the output file.	Change the input file or output file name.	LIBGEN
LIBRARY GENERATION COMPLETE.	Message issued when generation of a library is completed.	None.	LIBGEN
LIBRARY GENERATION FILE EMPTY.	The file to be processed is empty.	Verify that file is local to job and contains data.	LIBGEN

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
LINE NUMBER LIMIT EXCEEDED.	The line number encountered or required during a resequencing (RESEQ) operation exceeded 99999.	Examine program and correct line number in error.	RESEQ
LINE(S) TRUNCATED.	Informative message. File lfn1 contains lines longer than the maximum length processed by the FCOPY. These lines were truncated when written to lfn2.	None.	FCOPY
LISTLB ABORT.	A fatal error has occurred while processing the LISTLB control statement.	Refer to dayfile for cause of problem.	LISTLB
LISTLB COMPLETE.	Informative message indicating that the LISTLB operation has finished.	None.	LISTLB
LOCAL FILE LIMIT.	The user has too many local files.	Return one or more local files and retry.	LFM, QFM, IRO
LOCAL FILE LIMIT, AT addr.	The job's local file limit has been exceeded in an attempt to GET or ATTACH the file.	Return one or more local files.	PFM
LOCAL FILE LIMIT, FILE lfn AT addr.	The job's local file limit was exceeded in an attempt to define another file or attach an existing file to the job.	Return one or more local files.	CIO
LO72 COMPLETE.	Informative message indicating that the program has completed processing.	None.	LO72
LPxx,....	Refer to description of corresponding message beginning with EQ.		
LRxx,....	Refer to description of corresponding message beginning with EQ.		
LSxx,....	Refer to description of corresponding message beginning with EQ.		
LTxx,....	Refer to description of corresponding message beginning with EQ.		
M.T. NOT AVAILABLE ON FILE lfn AT addr.	The magnetic tape executive is not executing.	Inform site operator.	CIO
MAGNET NOT ACTIVE.	No UDT address or incorrect UDT address in FST or MAGNET not present.	Inform site analyst.	LFM, RESEX
MASS STORAGE DIRECTORY NOT WRITTEN.	On a GTR control statement, the user has requested that a mass storage directory record be written on a nonmass storage file.	Ensure that file resides on mass storage.	GTR

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
MASTER USER NUMBER REQUIRED.	Dayfile message indicating that the job did not enter a user number (via USER statement). This is needed for a master user list run and for a master user inquire run.	Rerun job with USER statement validation.	PROFILE
MESSAGE LIMIT.	The number of messages the job issued has exceeded the limit for which the user is validated. Message functions issued by compilers or applications programs that run at the user's job control point are also counted as user dayfile messages and thus are subject to the user's validated dayfile message limit.	Split job into two or more jobs and retry.	1AJ, 1MA
MFL LESS THAN ECS MINIMUM CM FL.	To use ECS the user must have a required minimum amount of CM FL. This message indicates user does not have the required CM FL.	Increase CM FL.	1MA
MFL REQUEST TOO SMALL, MINIMUM USED.	MFL request was less than CONTROL's RFL= value. CONTROL's RFL= value is used for this MFL request, thus allowing further MFL requests.	Check the description of field length control in section 3.	CONTROL
MISSING DEMAND FILE ENTRY.	Dayfile message indicating RESEX internal problem. The overcommitment algorithm was initiated without a demand file entry having been defined previously.	Inform site analyst.	RESEX
MISSING VSN OR EQUIPMENT ASSIGNMENT.	Dayfile message indicating internal malfunction in RESEX (expected VSN or equipment assignment was not found).	Inform site analyst.	RESEX
MONITOR CALL ERROR.	RA+1 call unrecognized.	Examine program to determine why illegal RA+1 call is being made.	1AJ
MTxx,ASSIGNED TO filenam, VSN=vsn.	The specified file name is assigned to the tape volume with the specified VSN that is mounted on tape drive MTxx (the seven-track tape unit with EST ordinal xx).	Informative message. Processing of the tape file can begin.	RESEX
MT,Ccc,Eec,Hhhhhhhhh,B.C.RESTART.	Magnetic tape controller controlware restarted.	None.	1MT
MT,Ccc,Eec,Hhhhhhhhh,BAD ERASE.	Error detected after an erase was attempted to recover a write error.	Inform site analyst.	1MT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
MT,Ccc,Eec,Hhhhhhhhh,BID RECOVERY-x.	A single block mispositioning error was recovered by block ID recovery. If x is B, the error was caused by backspacing the tape too far; if x is F, the tape was not backspaced far enough.	None.	1MT
MT,Ccc,Eec,Hhhhhhhhh,BLOCK TOO LARGE.	Data block is at least one byte longer than length bbbb shown in third line of message.	None.	1MT
MT,Ccc,Eec,Hhhhhhhhh,BUSY.	Unit was still busy after 1 second.	Inform customer engineer.	1MT
MT,Ccc,Eec,Hhhhhhhhh,CHANNEL ILL.	Channel is not accepting function for status requests properly.	Inform customer engineer.	1MT
MT,Ccc,Eec,Hhhhhhhhh,CON.REJ.	Connect reject; unable to connect to the unit.	Inform site analyst.	1MT
MT,Ccc,Eec,Hhhhhhhhh,CON REJ. MDI.	Connect reject; unable to connect to unit because of marginal detection indication (thermal warning). Unit turned off.	Inform customer engineer.	1MT
MT,Ccc,Eec,Hhhhhhhhh,CON.REJ.OFF.	Connect reject; unable to connect to unit. Unit turned off.	Inform site analyst.	1MT
MT,Ccc,Eee,Hhhhhhhhh,DENSITY CHANGE.	The tape subsystem detected a change in the data recording density on the tape. Error is due to a malfunctioning tape unit or a bad tape.	Ask customer engineer to examine the tape unit. If tape unit does not require maintenance, discard the tape.	1MT
MT,Ccc,Eec,Hhhhhhhhh,FNff,Pyyyy.	Function ff was rejected by the controller; yyyy is the address in 1MT where the function was initiated.	Inform site analyst.	1MT
MT,Ccc,Eec,Hhhhhhhhh,Lbbbb,Bnnnnnn.	The length (bbbb) and block number (nnnnnn) read from trailer bytes in block did not match the actual length or the block number read; given in previous message line.	None.	1MT
MT,Ccc,Eec,Hhhhhhhhh,LOAD CHECK.	Load sequence failed on the unit.	Push CLEAR button and reload tape or contact site analyst.	1MT
MT,Ccc,Eec,Hhhhhhhhh,MARGINAL DOWN.	Indicates controller failure. Channel has been logically turned off and maintenance is required.	Inform customer engineer.	1MT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
MT,Ccc,Eec,Hhhhhhhhh,MARGINAL OFF.	Unit has been logically turned off because of read/write failure. This occurred when a special function to check the read/write path to a unit failed during initial label scan. Maintenance is required.	Inform customer engineer.	1MT
MT,Ccc,Eec,Hhhhhhhhh,NO EOP.	No end-of-operation detected from unit within 1 second.	Inform customer engineer.	1MT
MT,Ccc,Eec,Hhhhhhhhh,NOISE.	A noise block was skipped on the tape.	None.	1MT
MT,Ccc,Eec,Hhhhhhhhh,NOT READY.	Tape unit dropped ready status.	Make unit ready.	1MT
MT,Ccc,Eec,Hhhhhhhhh,ON THE FLY.	Error was corrected as the data was read.	None.	1MT
MT,Ccc,Eec,Hhhhhhhhh,POSITION LOST.	The last good block written cannot be found during write recovery.	None.	1MT
MT,Ccc,Eec,Hhhhhhhhh,RECOVERED.	Previously reported error has been successfully recovered.	None.	1MT
MT,Ccc,Eec,Hhhhhhhhh,STATUS.	Error type cannot be determined so actual controller status is returned.	Inform site analyst.	1MT
MT,Ccc,Eec,Hhhhhhhhh,WRONG PARITY.	Tape was written in parity opposite that being read.	None.	1MT
MT,Ccc-e-uu,vsn,rw,xx,Ss,GSgggggggg. MT,Ccc,Dddd...d. MT,Ccc,Uuu...u,Ttttt. MT,Ccc,Fff,Iii,Bnnnnnn,Lbbbb,Pppppppppp. MT,Ccc,Eec,Hhhhhhhhh,type. or MT,Ccc-e-uu,vsn,rw,xx,Ss,GSgggggggg. MT,Ccc,Dddd...d. MT,Ccc,Fff,Iii,Bnnnnnn,Lbbbb,Pppppppppp. MT,Ccc,Eec,Hhhhhhhhh,type.	Four or five-line message describing a magnetic tape hardware malfunction on a 66x or 67x tape unit. Message as illustrated indicates 7-track, model 667 or 677 unit. If NT appears in place of MT, message indicates 9-track, model 669 or 679 unit. Message is issued to error log and dayfile. The first line of each message provides the following information. cc-e-uu — Channel, equipment (tape controller), and physical unit number of tape unit on which error was encountered. vsn — Volume serial number associated with tape on the specified unit. rw — Read (RD) or write (WR) operation; any operation not involving an actual read or write is listed as a read. xx — EST ordinal of the unit on which the tape was written. This is provided only for labeled tapes	Refer to the separate listing of the last line message (MT,...,type.) for the appropriate action.	1MT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE

generated under NOS; otherwise, the field is blank.

s — Channel status.

gggggggg — General status of magnetic tape unit. Last byte is block ID.

The MT,Ccc,Dddd...d line of the message provides the following information.

cc — Channel number; the channel number is repeated to allow the analyst to associate this message with the first message if errors are occurring on more than one tape channel at the same time.

ddd...d — Detailed status of magnetic tape unit.

The MT,Ccc,Uuu...u,Tttt line of the message provides the following information.

cc — Channel number; repeated to associate this message with the previous messages.

uu...u — Unit status of the magnetic tape unit.

tttt — Third byte of the tape unit format parameters (refer to the magnetic tape subsystem reference manual for descriptions of unit format parameter fields).

The MT,Ccc,Fff,...,Pppppppp line of the message provides the following information.

cc — Channel number; repeated to associate this message with the previous messages.

ff — Software function on which the error occurred.

ii — Error iteration; number of times error has been encountered on this unit without successful recovery.

nnnnnn — Block number on which error occurred.

bbbb — Length of block on which error occurred in octal bytes.

pppppppp — MT internal error parameters.

The last line of each message provides the

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	following information. cc — Channel number; repeated to associate this message with the previous messages. ec — Octal error code value. hhhhhhhh — Parameters passed to the tape unit for the format function (refer to the tape drive's hardware reference manual for descriptions of the unit format parameter fields). type — Additional description of the error. Refer to individual listing of the last line message.		
MT/NT CONFLICT.	Conflict exists between 7-track and 9-track tape descriptors (track type, density, and conversion mode). For example, a request for a 9-track tape specified 200-bpi density. This message can also be issued if the device type specified in FET+1 conflicts with the track type specified in FET+8, bit 56. If dt=MT and bit 56 is set, the message is issued.	Ensure accuracy of control statement.	RESEX, BLANK
MULTI-FILE NOT FOUND, lfn AT addr.	Either LISTLB has reached the end of the multifile set or the requested file was not found. The following additional messages are also given. REQUESTED SECTION xxxx. FOUND SECTION yyyy. or FILE IDENTIFIER NOT FOUND. The lfn, addr, xxxx, and yyyy given can be ignored.	If LISTLB reached the end of the multifile set, the operation is complete and no action is required. Otherwise, ensure that the correct tape is being used and that it contains the desired file(s). All label parameters must match in order to position to the specified file.	1MT
**** NEXT TIME USE *ROUTE* INSTEAD OF *DISPOSE*.	Warning message indicating that the DISPOSE control statement will not be availble in future versions of NOS.	Convert procedures to use the ROUTE control statement.	FILES
**** NEXT TIME USE *ROUTE* INSTEAD OF *SETID*.	Warning message indicating that the SETID control statement will not be available in future versions of NOS.	Use the ROUTE control statement to set the id of a file when queueing it.	FILES

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
NEXT VSN est,vsn	Tape volume vsn is mounted on tape unit est and accepted as the next volume in the file being processed.	None.	1MT
NO CONNECT TIME AVAILABLE.	Dayfile message indicating that the user has accumulated the maximum connect time allowed for the specified project number.	Contact installation personnel in order to increase maximum connect time allowed.	CHARGE
NO CPU TIME AVAILABLE.	Dayfile message indicating that the user has accumulated the maximum CPU time allowed for the specified project number.	Contact installation personnel in order to increase maximum CPU time allowed.	CHARGE
NO ECS.	A DMPECS or DMDECS control statement or DED or DEP request was entered and no ECS field length is assigned to the user.	None.	CPMEM
NO FILE FOUND - filenam.	The file specified in the READ directive of a submit file could not be found.	Ensure that the specified file exists.	SUBMIT
NO FILE(S) RELEASED.	No files of the type PRINT or PUNCH were found local to the user's control point.	Examine control statement record to determine if PRINT or PUNCH files were expected.	OUT
NO HDR1 LABEL RETURNED ON OPEN.	No HDR1 label was found in the label buffer after the OPEN function was completed. Indicates a possible system error.	Inform site analyst.	LISTLB
NO INPUT FILE FOUND.	No valid input file exists; functions cannot be performed.	Verify that input file is valid.	QFM
NO LINE NUMBER, FN=filenam.	Input line in file filenam does not contain a line number.	Correct and rerun.	MSORT
NO LINE NUMBER ON SORT FILE.	A line on the input file to a SORT request is missing a line number or a line exceeded the 150-character limit.	Check the format of the input file.	SORT
NO LINE TERMINATOR AT EOR(S).	The user attempted to copy a file in which the last line of one or more records did not have a line terminator.	Check the structure of the file to be copied.	COPYCF, COPYCR, COPYSBF
NO MASS STORAGE AVAILABLE.	User attempted to reserve a track for a null primary file.	Retry later.	LFM
NO PRIMARY FILE.	A control statement needed the primary file as the default file. No primary file was found.	Specify a file name on the control stahement or create a primary file with the NEW, OLD, or PRIMARY statement, and rerun the job.	PFILES

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
NO READ FILE FOUND - filenam.	The file specified on the /READ directive cannot be found.	Ensure that file name specified is correct and that the file is a local file or a permanent file.	SUBMIT
NO SOURCE FILE SPECIFIED.	No file name was specified on the control statement call.	Specify a file name on the SUBMIT control statement.	SUBMIT
NO WRITE ENABLE, ON lfn AT addr.	Either the user attempted to write on a tape mounted with no write ring or no write was allowed because of additional constraints described in an additional message line.	If no additional message line appears ensure the inserting of a write ring by specifying the W processing option on the tape request (for example, PO=W on the LABEL control statement. Otherwise, refer to the description of the message in the additional message line.	1MT
NON-MATCHING CONVERSION.	Informative message indicating conversion mode on labeled 9-track tape differs from that specified by assignment request. System writes tape in specified mode, or reads tape with write ring out in correct mode. However, reading tape with write ring in or using wrong conversion mode generates conversion errors.	If reading tape with write ring in, return and reassign with correct conversion mode.	RESEX
NON-MATCHING DENSITY.	Informative message indicating that the density specified on the control statement or macro is not the same as the density of the assigned tape. Issued only to 9-track tapes with write ring out. 9-track tapes are read at the current density on tape. They are written at specified density if write initiated from load point; otherwise, tape is written at the current density on the tape.	None.	RESEX
NORERUN/RERUN IGNORED FROM TTY JOBS.	User entered NORERUN or RERUN from a terminal. The command is ignored.	None.	QFM, CONTROL
NOTICE*** DATA READ ERROR - n WORDS LOST.	Read error caused loss of n words in the dayfile.	None.	DAYFILE

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
NOTICE*** FILE TOO LONG - n WORDS.	The dayfile is n words longer than the system file manager reported to DAYFILE.	Inform site analyst.	DAYFILE
NOTICE*** FILE TOO SHORT - n WORDS LOST.	The dayfile is n words shorter than the system file manager reported to DAYFILE.	Inform site analyst.	DAYFILE
NT....	Refer to description of MT.... series of messages.		
NT DENSITY CONFLICT.	9-track tape unit specified by EST ordinal on ASSIGN statement does not support the required density.	Ensure that density compatible tape unit is specified.	RESEX
NT DRIVE CONFLICT.	One of the following. 9-track tape unit specified by EST ordinal on ASSIGN statement conflicts with other resource requirements for this job. In this case the system rejects assignment to prevent the job from deadlocking itself. or Increased resource demands (RESOURC control statement) cannot be satisfied due to conflicts with currently assigned resources (job would deadlock itself).	Check the description of the RESOURC statement in section 6. Reduce resource demand which causes conflict.	RESEX
OLD MASTER FILE EMPTY OR MISPOSITIONED	None of the replacement file records were copied to the new file.	Ensure that the file is assigned to the job. If file is assigned, rewind the file.	COPYL, COPYLM
OLDPL ERROR.	Update program library format contains an error.	Correct error and rerun.	UPMOD
OPERATOR DROP.	Informative message indicating that operator dropped job.	None.	DSD, 1AJ
OPERATOR EVICT.	The operator entered an EVICT command to drop the job from the rollout queue. This disallows EXIT, EREXIT, and REPRIEVE processing.	Contact site operator to determine the cause of job eviction.	1RI
OPERATOR KILL.	The operator entered a KILL command to drop the job. This disallows EREXIT processing. A job with extended REPRIEVE processing is reprieved once. EXIT processing is allowed.	Correct job as needed and rerun.	1AJ
OPERATOR OVERRIDE.	The operator entered an OVERRIDE command to drop the job. This disallows EXIT, EREXIT, and REPRIEVE processing.	Correct job as needed and rerun.	1AJ

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
OUTPUT FILE LIMIT.	The total number of files disposed to the output queue by the job has exceeded the limit for which the user is validated.	If possible, split job into two or more jobs and retry. Otherwise, reduce number of files by copying output to single file and then issuing dispose.	LFM
OUTPUT FILE LIMIT, FILE lfn AT addr.	During an attempt to close this file, the number of files disposed to output queues by the job has exceeded the limit for which the user is validated.	If possible, split job into two or more jobs and retry. Otherwise, reduce number of files by copying output to single file and then issuing dispose.	CIO
OVERLAY FILE EMPTY.	No data appears in the requested file.	Verify that overlay file is valid.	1AJ
OVERLAY FILE NOT FOUND.	The specified file was not available.	Verify that file is local to job and retry.	1AJ
OVERLAY LOST.	The specified overlay was not found.	Verify that file with specified overlay is local to job.	1AJ
OVERLAY NOT FOUND IN LIBRARY - ovlname.	The overlay ovlnam was not found in the system library.	Verify that call is to valid overlay.	LDR
PACK PARAMETER ERROR.	The PACK control statement contains too many or no file names.	Check the format of the PACK control statement in section 7.	PACK
PARITY ERROR - RESTARTED FROM kk.	Because RESTART detected a parity error in attempting to restart from the specified checkpoint nn, the alternate checkpoint kk was used instead.	None.	RESTART
PF STAGING DISABLED, AT addr.	The specified direct access file resides on the Mass Storage Facility (MSF) and the site has temporarily disabled all MSF file staging.	Ask site operator when MSF file staging will resume and retry the job at that time.	PFM
PF UTILITY ACTIVE, AT addr.	Because a permanent file utility is currently active, the operation was not attempted; the user should retry the operation.	Wait until PF utility is not active and retry.	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
PFM ABORTED, AT addr.	Error flag detected at PFM control point.	Inform site analyst.	PFM
PFM ILLEGAL REQUEST, AT addr.	One of the following. - Illegal command code passed to PFM - Illegal permit mode or catalog type specified - CATLIST request has permit specified without a file name - PERMIT command or macro attempted on a public file	Verify that PFM request is valid.	PFM
POSITION ERROR ON-filenam.	File filenam was not repositioned after being checkpointed because CHKPT detected an address error.	None.	CHKPT
POSITION LOST, lfn AT addr.	During write or read error recovery, the system could not find the last good block of data, making it impossible to successfully perform error recovery. Labels are not written after this error and existing data on the tape is not destroyed.	Retry operation on different tape and/or tape drive, if possible.	1MT
PP CALL ERROR.	The monitor detected an error in a CPU request for PP action.	Verify that correct PP call is issued.	1AJ
PRIOR TAPE ASSIGNMENT LOST.	Magnetic tape executive has been dropped along with tapes assigned. All of the user's prior tape assignments are lost.	Terminal user must return/unload all prior tapes and reassign. Batch user is aborted and must rerun his job.	RESEX
PROCEDURE FILE EMPTY.	Procedure file specified contains no data.	Verify that procedure file contains data and retry.	CONTROL
PROCESSING OPTION NOT APPLICABLE.	The processing options, PO=R and PO=D, are allowed only on a COPY statement copying from mass storage, or an I or SI-binary format tape to an S or L format tape. The processing option, PO=T, is allowed on a TCOPY statement that generates an E or B format tape.	Refer to the COPY or TCOPY statement description in section 7, correct the statement, and retry.	COPY, TCOPY
PROFILE FILE CREATE COMPLETE.	Dayfile message indicating that the creation run is complete.	None.	PROFILE
PROFILE FILE DATA BASE ERROR.	Dayfile message indicating that the project file does not contain both a level 0 and level 1 block.	Ensure that the project file is local and contains a level 0 and level 1 block (at least	PROFILE

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
		one charge entry) and rerun.	
PROFILE FILE INQUIRY COMPLETE.	Dayfile message indicating that the inquire run is complete.	None.	PROFILE
PROFILE FILE LIST COMPLETE.	Dayfile message indicating that the list of PROFILa is complete.	None.	PROFILE
PROFILE FILE REFORMAT COMPLETE.	Dayfile message indicating that the reformat run is complete.	None.	PROFILE
PROFILE FILE SOURCE COMPLETE.	Dayfile message indicating that the source run is complete.	None.	PROFILE
PROFILE FILE UPDATE COMPLETE.	Dayfile message indicating that the update run is complete.	None.	PROFILE
PROGRAM NOT FOUND.	The program to be loaded was not found on the specified library file.	Verify that the program is on library file.	EXU
PROGRAM NOT ON MASS STORAGE.	The program does not reside on a mass storage device.	Copy program to mass storage.	EXU
PROGRAM STOP.	The system processed a program stop (00) instruction.	None.	1AJ
PROGRAM STOP AT addr.	The monitor detected a program stop instruction at address addr.	None.	1AJ
PROGRAM TOO LONG.	The program does not fit in the available storage.	Increase field length and retry.	EXU
PROJECT NUMBER EXPIRED.	Dayfile and output file message indicating that the project number expiration date has occurred.	None.	CHARGE
PROTECTED FILE.	The user has attempted to release a locked file.	None.	LFM
PRU LIMIT, AT addr.	The job's mass storage PRU limit was exceeded during preparation of a local copy of an indirect access file.	Return one or more local files and retry.	PFM
PRU LIMIT, FILE lfn AT addr.	The job's mass stoage PRU limit was exceeded during an attempt to write or extend this file.	Return one or more local files and retry.	CIO
PRUS REQUESTED UNAVAILABLE, AT addr.	The number of PRUs specified via the S parameter on the DEFINE request is not available.	Request smaller number of PRUs.	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
QAP - BUFFER ARGUMENT ERROR.	Dayfile message indicating one of the following. - Buffer does not contain the required number of words of data for the particular function. - Buffer pointer (FIRST, IN, OUT, LIMIT) is out of range.	Verify that the calling program has set up the request correctly.	QAP
QAP - ILLEGAL USER ACCESS.	The user tried to perform an operation for which he is not validated (for example, attempting to run a system origin job from nonsystem origin).	Ensure accuracy of control statement or determine proper validation requirements.	QAP
QFM FILE EMPTY.	Either an empty file was specified on a READ directive in the submit source file or the NR parameter was specified on the SUBMIT statement so that the specified file was not rewound.	Check that the correct file is specified. If it is, rewind the file or remove the NR parameter from the SUBMIT statement.	QFM
QFM FILE NAME ERROR.	The lfn specified is not a valid file name.	Verify file name.	QFM
QFM FILE NOT FOUND.	The file to be submitted could not be found.	Verify that submit file exists.	QFM
QFM FILE NOT ON MASS STORAGE.	The file to be submitted does not reside on mass storage.	Copy file to be submitted to mass storage and retry.	QFM
QFM ILLEGAL FILE TYPE.	The file to be submitted is not a local file type.	Copy the submit file to a local file.	QFM
RA.SSC OUT OF RANGE.	The subsystem receiving the buffer pointer (RA.SSC) word has invalid data fields.	Correct RA.SSC word fields.	1MA, 1MB
READ AFTER WRITE, lfn AT addr.	The user attempted to read a tape on which the last operation was a write.	Ensure accuracy of tape positioning statements (BKSP, BKSPRU, SKIPFB, or REWIND required to read after write).	1MT
READ FILE BUSY - filenam.	The read file is found to be busy (direct access file only).	Retry after file is not busy.	SUBMIT
READY DROP, lfn AT addr.	Unit dropped ready.	None.	1MT
RECORD SIZE EXCEEDS 500.	The maximum line length for a record to be converted (500 characters) was exceeded.	Split lines that are too long into two or more lines.	CONVERT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
RECORD TOO LARGE ON filenam.	An input record was encountered that exceeded S or L output tape block size.	Reduce input record size, or use COPY control statement to increase S or L tape block size or allow record splitting and retry with PO parameter.	COPY, COPYBF, COPYEI
RECORD TOO LONG.	The record is too long for available memory. In response to a WBR request, the record length parameter was greater than or equal to the user's field length. Available memory is filled and the excess data is skipped.	Increase field length and rerun.	CPMEM
REMOVABLE PACKS OVERCOMMITMENT.	Removable pack request without NA selected causes resource overcommitment.	Retry later or retry with NA parameter on ATTACH, DEFINE, etc., with PN parameter specified.	RESEX
RENAME OF PROC TYPE NOT ALLOWED.	A RENAME of a PROC type record was attempted; this is not allowed.	None.	LIBEDIT
REPRIEVE BLOCK ERROR.	An address is out of range or there is an illegal parameter in the reprieve parameter block at the time of an error. The message is also issued if the specified reprieve address itself is out of range. (1AJ issues this message for all errors except terminal interrupts processed by 1RI.)	Ensure parameter block is correct.	1AJ, 1RI
REPRIEVE CHECKSUM BAD.	The computed checksum does not agree with the checksum specified in the parameter block at the time of the error. (1AJ issues this message for all errors except terminal interrupts processed by 1RI.)	Ensure interrupt handler is still intact. Ensure that code in the area for which checksum was computed has not changed.	1AJ, 1RI
REQUEST UNDEFINED ON DEVICE 1fn AT addr.	The specified function cannot be performed on the device on which the file resides. The system provides a dump of the FET on file OUTPUT.	Verify that valid device is specified.	CIO
RERUN NOT POSSIBLE.	The job cannot be rerun because of one of the following. - Job is time-sharing origin. - No input file is found for the job. - An error occurred in reading or writing the input file system sector.	None.	1DS

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	- Rerun status is disabled.		
RESEQ CONTROL CARD ERROR.	The RESEQ control statement contains a syntax error.	Correct error and rerun.	RESEQ
RESEQ ERRORS.	A resequencing error was encountered.	Refer to preceding message for more specific information about the error.	RESEQ
RESEQ NUMERIC PARAM ERROR.	A parameter which is supposed to be numeric contains a nonnumeric character.	Correct error and rerun.	RESEQ
RESERVED FILE NAME.	A reserved file name was incorrectly used.	Choose a nonreserved file name.	OPLEDIT, EDIT, DATADEF, IAFEX, TELEX
RESEX ABORT - OPERATOR TERMINATION.	The operator entered a DROP, KILL or RERUN command, setting an error flag in RESEX. RESEX performed appropriate cleanup procedures before termination.	Determine reason for operator action. Rerun job, if desired.	RESEX
RESEX ABORT - SYSTEM RESOURCE LIMIT.	RESEX terminated prematurely due to job time limit, SRU limit, or track limit. RESEX performs appropriate cleanup procedures before termination.	If error caused by SRU or time limit, increase resource limits. If caused by track limit, contact site analyst.	RESEX
RESEX ABORT - TERMINAL INTERRUPT.	Terminal user interrupted RESEX (interrupt or terminate sequence). RESEX performs appropriate cleanup procedures before termination.	None.	RESEX
RESEX DETECTED ERROR.	The resource executive (RESEX) detected an error.	Inform site analyst.	LFM
RESEX FAILURE, AT addr.	The resource executive (RESEX) has detected a fatal error.	Inform site analyst.	PFM
RESOURCE DEMAND ERROR.	The user attempted to decrease the number of scheduled units to less than the number of currently assigned units or increase the number of scheduled units to a point where a deadlock would occur.	Adjust RESOURC statement parameters accordingly.	RESEX
RESOURCE ENVIRONMENT ERROR.	Dayfile message indicating RESEX internal problem occurred (internal environment building failed due to MST, UDT, or EST errors).	Inform site analyst.	RESEX

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
RESOURCE NEGATIVE SHARE COUNT.	Dayfile message indicating RESEX internal problem occurred. The resource overcommitment algorithm indicates a greater number of users are supposedly sharing a removable pack than are actually sharing the pack.	Inform site analyst.	RESEX
RESOURCE PF ERROR ec filenam.	PFM error ec occurred when attaching resource file filenam.	Inform site analyst.	RESEX
RESOURCE SCRATCH FILE ERROR.	Dayfile message indicating RESEX internal problem has occurred. An empty entry has been found in the overcommitment algorithm scratch file.	Inform site analyst.	RESEX
RESOURCE TYPE ERROR.	The user specified an illegal resource type.	Ensure accuracy of control statement.	RESEX
RFL BEYOND MFL.	The RFL request is greater than the maximum field length for a job step.	Increase maximum field length with MFL statement.	CPM
ROLLIN FILE BAD.	A job could not be rolled in correctly.	Inform site analyst. Check error log dayfile for the job that was aborted and the location of the bad rollin file.	IRI
ROUTE COMPLETE.	Informative message indicating ROUTE operation has completed.	None.	ROUTE
ROUTE COMPLETE. JOB NAME IS jobnam.	A ROUTE statement entered a file in the input queue. The routed job has the system job name jobnam.	None.	ROUTE
ROUTE CONTROL CARD ERROR.	Format of the control statement is incorrect.	Check the ROUTE statement format in section 7.	ROUTE
ROUTE *DC* INCOMPATIBLE WITH *EC*.	The user specified a DC/EC combination that is not legal.	If the DC parameter implies a print or punch file, the EC parameter must be for the same file type.	ROUTE
ROUTE *FID* IGNORED.	Informative message listed for NOS/BE compatibility.	None.	ROUTE
ROUTE ILLEGAL KEYWORD.	Control statement contains a duplicate or illegal keyword.	Check the ROUTE statement format in section 7.	ROUTE

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ROUTE ILLEGAL *OT* PARAMETER.	The origin type specified by the OT parameter is illegal.	Verify that correct origin type is specified.	ROUTE
ROUTE *OT* NOT ALLOWED.	The user program is not system origin. Only system origin jobs can use the OT parameter.	The user without system origin privileges should not attempt to use the OT parameter.	ROUTE
ROUTE *PRI* IGNORED.	Informative message listed for NOS/BE compatibility.	None.	ROUTE
ROUTE *REP* GT 31. DEFAULT USED.	The repeat count specified was greater than 31; it has been set to 0. This condition does not abort the program.	Use a repeat count less than 32.	ROUTE
ROUTE *ST* IGNORED.	Informative message listed for NOS/BE compatibility.	None.	ROUTE
ROUTE *TID* AND *FM/UN* CONFLICT.	The TID parameter was specified with either the FM or UN parameter. Either one of these parameters is mutually exclusive with TID.	Refer to section 7 for the correct format of the ROUTE control statement.	ROUTE
ROUTE *TID/FM/UN* and *ID* CONFLICT.	The ID parameter was specified with the TID, FM, or UN parameter.	Refer to section 7 for the correct format of the ROUTE control statement.	ROUTE
ROUTE *TID=xx. VALUE IGNORED.	Informative message listed for NOS/BE compatibility.	None.	ROUTE
RTIME nnnnnn.nnn SECS.	Dayfile message output by RTIME control statement giving the real-time seconds count.	None.	IAJ
SECURE MEMORY, DUMP DISABLED.	An attempt was made to dump memory protected by the system. This message is also issued when a user enters a DMD or DMP statement at a terminal if the statements are not part of a procedure.	Refer to Security Control in section 3 and the DMD and DMP statement descriptions in section 9 of the NOS Reference Manual, volume 1, or to Security Considerations in section 2 of the NOS Reference Manual, volume 2. To obtain memory dumps from a time-sharing job, include the DMD or DMP statement in a procedure or in an ENTER statement. A DMD	1AJ

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
		statement can be included only in a batch job.	
SFP CALL ERROR.	Dayfile message indicating that SFP was called directly and not by PLL.	Inform site analyst.	SFP
SFP/D00 BAD ERROR TEXT.	The specified systems text is not in the correct format.	Rebuild systems text using correct format.	SFP
SFP/D00 ERROR TEXT NOT FOUND.	The specified systems text could not be found.	Ensure the systems text specified is correctly installed on the system or call site analyst.	SFP
SFP/D00 ERROR TEXT NOT ON MASS STORAGE.	The specified systems text does not reside on mass storage.	Move systems text to mass storage.	SFP
SFP/D00 INVALID MESSAGE NUMBER.	The specified message number could not be found on the systems text.	Ensure that D00 was called with the correct message number.	SFP
SFP/xxx ILLEGAL FUNCTION.	The function code in the RA+1 call is in error.	Correct and retry.	SFP
SFP/xxx ILLEGAL FUNCTION CODE.	The specified function code to the program xxx is not defined.	Ensure the correct function code was used.	SFP
SFP/xxx ILLEGAL ORIGIN CODE.	Dayfile message indicating that the function was illegal for the user's origin.	Inform site analyst.	SFP
SFP/xxx PARAMETER ERROR.	Dayfile message indicating that the parameter address was outside the field length.	Inform site analyst.	SFP
SFP/PFE I/O SEQUENCE ERROR.	Action was requested on a file that was already busy.	Wait until the file is not busy and retry.	SFP
SFP/RPV UNABLE TO RESET, NOT REPRIEVED.	Dayfile message indicating that an attempt was made to reset when the job had not been reprieved.	Inform site analyst.	SFP
SFP/SRP SPECIAL REQUEST PROCESSING ERROR.	Dayfile message indicating that the SPCW word was busy.	Inform site analyst.	SFP
SFP/STS UNKNOWN DEVICE TYPE/NAME.	The device code is not recognized by the system.	Check device code for validity.	SFP
SHARE TABLE ERROR.	Dayfile message indicating RESEX internal problem occurred (share table full or matching entry not found).	Inform site analyst.	RESEX

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
SL NOT VALIDATED.	The SRU limit requested exceeds that for which the user is validated.	Request smaller SRU limit.	CPM
SPCW CALL ERROR.	A DMP= type call was made, and the program called is either not in the CLD or does not have a DMP= entry point defined.	Inform site analyst.	1AJ
STATUS ERROR, lfn AT addr.	An irrecoverable error was encountered. A second message line describes the error in more detail.	Retry or inform site analyst.	1MT
	ARA BURST DEFECTIVE. — First block of tape at 1600 or 6250 cpi cannot be read or written. Another unit or tape should be tried.		
	CRC ERROR. — An error was detected in cyclic redundancy character.		
	FILL STATUS ILLEGAL. — The system has detected an odd number of frame, a condition which is illegal for the data format of the tape being read.		
	MULTI-TRACK PHASE ERROR. — Multiple tracks were found to be in error at 1600 cpi, making recovery impossible.		
	PARITY ERROR. — The tape could not be read/written correctly.		
	UNIT HAS MOTION PROBLEMS. — The tape unit cannot properly write the tape. The user should resubmit his job, using a different tape unit.		
	UNIT PROBLEMS. — Unit check bit is set in detailed status (67x units only). The user should try another unit.		
	POSTAMBLE ERROR. — A missing or defective postamble was detected at 1600 cpi.		
	SINGLE FRAME ERROR. — A frame (NRZI only) containing all zeros was read; data will be at least		

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	one frame short.		
	LRC ERROR. The longitudinal redundancy check character was read incorrectly (9-track NRZI).		
	ILLEGAL CHARACTER. Illegal character read from 9-track tape. If a 1 is detected in bit 6 of a translated character, the character is illegal.		
STEP CONDITION.	Step mode flag set in the PSD register caused the program to interrupt at the end of a program instruction with an exchange jump to EEA (the error exit address in the exchange package).	Inform site analyst.	1AJ
SUBMIT COMPLETE. JOBNAME IS jobname	The SUBMIT statement successfully entered the file in the input queue. The system's job name for the entered job is jobname.	None.	SUBMIT
SUBMIT FILE EMPTY.	An EOR or EOF was encountered on the submit file before any data was found.	Rewind submit file, verify format of data and that the file is a local file, and retry operation.	SUBMIT
SUBSYSTEM ABORTED.	The user job was connected (either long term connection or wait response set) to a subsystem which aborted.	Retry later.	1AJ
SYSTEM ABORT.	Possible causes include the following. - Invalid USER statement - Attempt to access a restricted subsystem - Operator evicted job - Unrecognizable error flag - SSJ= block outside field length - 1RI detected a bad rollout file - 1RO detected an unrecoverable ECS parity error during rollout of a job's ECS field length.	If the cause was an invalid USER statement or an attempt to access a restricted subsystem, correct the job and rerun. Otherwise, contact a site analyst.	1AJ
SYSTEM CHECKPOINT ABORT.	The checkpoint system was up and aborted, and in the process aborted a number of subsystems.	Informative message.	1CK
SYSTEM ERROR.	One of the following. - RESEX cannot communicate wtih subsystem via the RSB or SIC monitor calls. - LFM cannot complete the requested LFM function because the calling program has	Inform site analyst.	BLANK, LFM, PFM, RESEX

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	a DMP= entry point. - BLANK is unable to read low core pointers via the RSB monitor call. - PFM detects error condition.		
TABLE OVERFLOW.	The job field length is too small to hold the tables for processing the GTR statement.	Increase field length and rerun job.	GTR
TAPE BLANK LABELED.	Dayfile message indicating that the blank label operation successfully completed.	None.	BLANK
TAPE BLOCK DEFINITION ERROR.	The user attempted to define data block size via the FC or C keyword or noise block size via the NS keyword in such a manner that the system is unable to correctly define the size of the data block. The omission of the FC or C parameter on a control statement where it is required also causes this message to be issued.	Ensure accuracy of control statement.	RESEX
TAPE FORMAT PROBABLY WRONG.	This message is issued in addition to one of the following messages. BLOCK SEQUENCE ERROR, lfn AT addr. BLOCK TOO LARGE, lfn AT addr. WRONG PARITY, lfn AT addr.	Ensure accuracy of format (F) parameter on control statement or macro.	1MT
TIME LIMIT.	The execution time limit for a job step expired resulting in job termination.	If a time limit was set for the job, include a SETTL statement requesting a longer time limit for the job step. If the job step time limit was the maximum for which the user is validated, request a larger time limit or decrease the amount of processing to be performed by the job step.	1AJ
TL NOT VALIDATED.	The time limit requested exceeds that for which the user is validated.	Request smaller time limit.	CPM, ACCFAM
TOO MANY ARGUMENTS.	The number of arguments on the control statement exceeds that allowed by the program.	Reformat the control statement with an allowable number of arguments.	SUBMIT, TCS

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
TOO MANY DEFERRED BATCH JOBS.	The user is not validated for this function or he has more jobs in the system than he is allowed. (All jobs in local and remote batch queues are counted.) The count is ignored if the job is of system origin or the user is validated for system privileges and DEBUG mode is set by the operator.	Resubmit the job when a queued deferred batch job has been processed. To change your validation limit, contact site personnel.	QFM
TOO MANY FILES - NOT ALL PROCESSED.	The job had more files than could be processed.	Issue a series of REWIND statements or commands specifying file names. Inform the site analyst if this error occurs frequently.	MFILES
TOO MANY PARAMETERS.	More parameters were entered (including null parameters) than are allowed for command.	Ensure accuracy of entry.	GTR IAFEX NOTE TELEX
TOTAL ASSIGNED COUNT ERROR.	Dayfile message indicating RESEX internal problem (sum of individual resource assigned counts differs from total assigned count in demand file entry).	Inform site analyst.	RESEX
TOTAL DEMAND COUNT ERROR.	Dayfile message indicating RESEX internal problem occurred (sum of individual resource demand counts differs from total demand count in demand file entry).	Inform site analyst.	RESEX
TRACK LIMIT, FILE lfn AT addr.	The device on which the file resides is full.	Specify another device or contact site analyst.	CIO
TRAILER BLOCK COUNT ERROR, lfn AT addr.	The block count in the EOF1 or EOV1 label did not match the block count maintained by the tape executive during the read operation.	Inform site analyst.	1MT
UNABLE TO READ IQFT FILE.	An attempt to initialize inactive queues failed because the IQFT file could not be read.	Retry operation.	IMS, MSI
UNIDENTIFIED PROGRAM FORMAT.	The file the user requested to be loaded was in an unrecognizable format.	Check the format of the file.	1AJ
UNIT HUNG UP ON EOP OR BUSY, lfn AT addr.	Unit did not receive EOP on unit busy.	Inform site analyst.	1MT
UNLABELED TAPE REQUIRED - filenam.	An S tape used for E, B, or X tape conversion must be unlabeled.	Use unlabeled tape.	TCOPY
UNRECOGNIZABLE TYPE SPECIFIED.	The type of the source file was not legal.	Check type of source file.	L072

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
UNRECOGNIZED BACKSPACE CODE.	The specified backspace code was not 0, 1, 2, or 3.	Refer to the COPYX control statement description, correct the error, and retry.	COPYX
UNRECOVERABLE ERROR ON filenam.	An unrecoverable error such as wrong parity, density change, or ready drop was detected on the input file.	Either the tape was bad or the tape drive is malfunctioning.	COPY, TCOPY
UNRECOVERED MASS STORAGE ERROR.	A control statement or overlay calls a system library program. During the load, an unrecoverable read error occurred.	Inform site analyst.	1AJ
UNUSUAL END-OF-FILE ENCOUNTERED.	GTR detected an EOF mark not preceded by an EOR mark.	Dump file to determine file format error.	GTR
UPDATE COMPLETE.	The user's password has been changed.	None.	PASSWOR
UPDATED - type/name	The record with type and name on the old file was replaced with the matching record from the replacement file.	None.	COPYL, COPYLM
UPMOD COMPLETE.	Message indicating UPMOD completion.	None.	UPMOD
USER DAYFILE DUMPED.	The user dayfile dump is complete.	None.	DAYFILE
USER NOT VALIDATED FOR ECS MINIMUM CM.	The user was validated for the ECS specified on the job statement but was not validated for at least 10000 octal words of memory.	Request a larger CM validation limit from site personnel.	ACCFAM
VERIFY ERRORS.	Verification errors were encountered during comparison of date/records/files.	Check output file for errors.	VERIFY
VERIFY FL ABOVE USER LIMIT.	Field length for L or F tape verify exceeds user's current maximum.	Use MFL to increase maximum field length or reduce block size on L or F tapes.	VERIFY
VSN FILE ERROR.	Dayfle message indicating RESEX internal problem occurred (VSN file entry does not match job identification).	Inform site analyst.	RESEX

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
WRITE ON READ-ONLY FILE lfn AT addr.	Either the user attempted to write on a file with write interlock or the direct access file was not attached in WRITE mode.	Reattach file in write mode or clear write interlock.	CIO, IAFEX, TELEX
WRITE OVER LABEL.	The user attempted to write over the VOL1 label.	Have the operator blank label the tape.	1MT
WRITE OVER LABEL ILLEGAL ON lfn AT addr.	The user is not allowed to destroy the VOL1 label.	Use LISTLB control statement to obtain label data.	1MT
WRONG PARITY, lfn AT addr.	A 7-track tape is being read in opposite parity from which it was written.	Ensure accuracy of format parameter (F) on control statement or macro.	1MT
XL BUFFER/FET PARAMETER ERROR, lfn AT addr.	One of the following: - HDR1 label in extended label buffer or FET contains a nonnumeric display code value in a numeric field. - Character count in header word preceding labels in the extended label buffer does not equal 80.	Correct condition that caused error and retry.	1MT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
200BPI WRITE ILLEGAL.	The tape unit (667 or 677) cannot record data at 200 bpi.	Specify a different tape density.	1MT
25555 FIELD LENGTH INCREASE.	The job field length was too small for LIBEDIT. Field length was increased to 26K.	None.	LIBEDIT
filenam ALREADY PERMANENT, AT addr.	The user has already saved or defined a file with the name specified.	Save or define file using different file name or purge existing file.	PFM
jobnam ASSIGNS EXCEED DEMANDS.	Dayfile message indicating that RESEX internal problems occurred. The resources actually assigned to the job jobnam exceed the resources demanded on a RESOURC statement.	Inform site analyst.	RESEX
n BAD FORMAT BLOCKS.	Block(s) not meeting X or SI-coded conversion requirements have been encountered on input.	Check tape format for errors.	TCOPY
pfn BUSY, AT addr.	The specified direct access file is attached in an incompatible mode, or it is currently being accessed by one of the following. - More than 77B users in READ mode - More than 77B users in READAP mode - More than 7777B users in READMD mode	Reissue ATTACH until file becomes available, or issue ATTACH specifying NA option.	PFM
n DIRECTIVE ERRORS.	LIBEDIT could not interpret n directives because the records specified in the directives could not be found.	Correct errors as listed in LIBEDIT output and rerun job.	LIBEDIT
lfn EMPTY, AT addr.	The file specified on a SAVE request contains no data.	Verify that file contains data and retry.	PFM
nnn FILE(S) PROCESSED.	The operation was performed on nnn files.	None.	MFILES
xx FILE(S) RELEASED.	An xx number (octal count) of PRINT and/or PUNCH files were found and released from the user's control point.	None.	OUT
pfn FOUND. or un FOUND.	The file name specified on the CATLIST (LO=0) statement was found in the user's catalog or the file name or user number specified on the CATLIST (LO=P) statement was granted permission.	None.	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
pfn ILLEGAL FILE TYPE, AT addr.	The operation requested cannot be performed on the specified file because it is of the wrong file type. This message is issued when the user attempts to define a direct access file with the same name as a file currently assigned to the job that is of a file type other than local (LOFT).	Define the file using a unique name or return the conflicting file.	PFM
pfn IS DIRECT ACCESS, AT addr.	An indirect access file operation was attempted on a direct access file.	Use the appropriate direct access file request.	PFM
pfn IS INDIRECT ACCESS, AT addr.	A direct access file operation was attempted on an indirect access file.	Use the appropriate indirect access file request.	PFM
nnnn LINES TRUNCATED.	This message informs the user that nnnn lines were truncated because they were longer than 150 characters.	Determine if relevant data was lost in the truncation. If possible, split the too long lines and repeat the copy operation.	COPYCF, COPYCR, COPYSBF
jobnam MISSING RESOURCE.	Dayfile message indicating that RESEX internal problem occurred. RESEX expected but did not find a resource unit assigned to the specified job. This could occur if MAGNET was stopped while tapes were assigned.	Inform site analyst.	RESEX
n NOISE BLOCKS DELETED. n NOISE BLOCKS PADDED.	Block(s) on S or L output tape smaller than noise size have been deleted/padded.	None.	COPY
filenam NOT DECLARED RANDOM.	An EOF was encountered on the nonrandom file, filenam.	Verify that file name is in correct format.	LIBEDIT
filenam NOT FOUND.	RESTART was unable to retrieve a file named, but not included, on filenam.	Verify that filenam is valid.	RESTART
proc NOT FOUND.	Procedure specified in CALL statement cannot be found.	Verify that procedure name is correct and retry.	CONTROL
filenam NOT FOUND, AT addr. or un NOT FOUND, AT addr.	One of the following. - The specified permanent file could not be found. - The specified user number could not be found. - The user is not allowed to access the specified file. - The specified local file could not be	Verify that file name/ user number is correct, that access permission has been granted, and that correct access is being attempted.	PFM

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	found or was an execute-only file.		
xxx NOT IN PP LIB.	Dayfile message indicating that PP package xxx was not found in PP libraries.	Ensure that the correct PP package name was specified.	SFP
xxx NOT IN PP LIB. CALLED BY yyy.	Dayfile message indicating that PP package xxx, which was called by package yyy, was not found in the PP libraries.	Ensure that the correct PP package name was specified or inform site analyst.	SFP
filenam NOT ON MASS STORAGE, AT addr.	The file to be saved is not on mass storage; the first track of the file is not recognizable.	Verify that file is on mass storage.	PFM
n PARITY/BLOCK TOO LARGE ERRORS.	Parity and/or block-too-large errors have been encountered on the input file during the copy operation.	If dayfile shows block-too-large errors have occurred and tape is S, L, or F format, increase block size and retry; otherwise, tape is probably assigned in the wrong format. If parity errors have occurred, the tape is bad and the data on it cannot be correctly recovered.	COPY, TCOPY
pfn PERMANENT ERROR, AT addr.	The specified direct access file resides on the Mass Storage Facility (MSF) and has data errors that cannot be corrected.	Ask site analyst about file recovery from a backup copy.	PFM
nnnnn RECORDS CONVERTED.	Informative message indicating the number of records (nnnnn) converted from one character set to another.	None.	CONVERT
n RECORDS NOT REPLACED.	Informative message. LIBEDIT encountered n records on a replacement file that were not named in the directives and did not replace old file records.	Either change the directives so that the replacement file is a no-replace file or include an *IGNORE directive listing the records that are not to be used.	LIBEDIT
n RECORDS SPLITS OCCURRED.	Multiple blocks per record have been written on an S or L output tape.	None.	COPY

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
jobnam RESTARTED FROM yy/mm/dd. hh.mm.ss.	The checkpoint job identified by jobnam was restarted from the checkpoint taken on the specified date and time. This message is issued whenever a checkpoint job is restarted.	None.	RESTART
jobnam SHARE TABLE MISMATCH.	Dayfile message indicating that RESEX internal problem occurred. While processing the specified job, an expected share table entry match with the environment did not occur.	Inform site analyst.	RESEX

GLOSSARY C

Alphanumeric	The letters of the alphabet (A through Z) and the digits (0 through 9).
ASCII	American National Standard Code for Information Interchange. The standard character set code and set used for information interchange between systems. ASCII is the default code set for nine-track coded tapes.
Block	Data recorded between magnetic tape interrecord gaps; a tape PRU. Blocking groups user records for greater transfer efficiency.
BOI	Beginning-of-information; the point in a file before which no file data exists.
Byte	One of the five groups of 12 bits within a 60-bit central memory word.
Character	A symbol whose coded representation may be processed within the system. Refer to definitions of alphanumeric, graphic, and control character.
Checkpoint file	File on which the results of a partially completed job are dumped when a CKP control statement is processed.
CIO	Combined Input/Output. System routine that performs NOS I/O.
Control character	A character whose occurrence within a print file causes an action. Examples are the carriage return and line feed characters.
Control statement	A sequence of words and characters that call a system routine to perform a job step. The control statement must conform to format specifications and end with either a period or a right parenthesis.
Control statement record	1. A record containing control statements, such as a procedure file. 2. The first record of a job containing the sequence of control statements that specifies all steps for job execution.
CYBER Loader	The NOS product that prepares programs for execution by placing program instruction and data blocks in central memory.

CYBER Record Manager (CRM)	The NOS product that acts as an I/O interface between CIO and other NOS products including ALGOL 4, ALGOL 5, COBOL 5, DMS-170, FORTRAN Extended 4, FORTRAN 5, and PL/I. The CRM functions are split between two file processors, Basic Access Methods (BAM) and Advanced Access Methods (AAM). Refer to the CRM manuals listed in the preface for definitions of CRM file organizations, block types, and record types.
Deadstart	The process of initializing the system by loading the operating system programs and the product set programs from magnetic tape or disk. Deadstart recovery is reinitialization after system failure.
Direct access file	A permanent mass storage file that can be assigned to a user's job. All changes to this file are made on the file itself rather than a working copy of the file (refer to definition of indirect access file).
Display code	A 6-bit code set used to represent characters in central memory and in binary files. Appendix A translates the code set.
EBCDIC	Extended Binary Coded Decimal Interchange Code. An 8-bit code set that, if the user requests, NOS uses to read or record data on nine-track tapes.
Empty PRU/record	Refer to definition of zero-length PRU.
Entry point	A location within a program that can be referenced by name from other programs.
EOF	End-of-file indicator; the file may be part of a multifile file whose end is defined by an EOI indicator. In the product set manuals an end-of-file may be called an end-of-partition.
EOI	End-of-information indicator; marks the end of a named file.
EOR	End-of-record indicator; marks the end of a logical record. In the product set manuals an end-of-record may be called an end-of-section.
EOT	End-of-tape; metallic strip marking the end of the recordable portion of a magnetic tape.
FET	File environment table; a table used by a COMPASS programmer to define and interrogate the current status and properties of a file assigned to a job.
File	1. Data that begins at BOI and ends at an EOI and is referenced by a one- to seven-alphanumeric-character name. 2. A portion of a multifile file ending with an EOF. 3. Data recorded on a magnetic tape beginning after an HDR1 label and ending before an EOF1 label.

NOS control statements requiring a file name lfn (except tape assignment statements) refer to definition 1; NOS control statements that have a parameter specifying the number of files refer to definition 2. Definition 3 applies only to labeled magnetic tapes.

File set

One or more tape files referred to by the lfn on a tape assignment statement. A file set may consist of:

- One file recorded on a single volume.
- More than one file recorded on a single volume.
- One file recorded on more than one volume.
- More than one file recorded on more than one volume.

All files within a file set have the same set identifier in their HDR1 labels.

File type

A category of files handled similarly by the system. Section 2 categorizes NOS files according to permanent file types and types of files assigned to user jobs.

FIT

File information table. An extension of a FET that defines a file for access by CYBER Record Manager.

FNT/FST entry

File name table/file status table (FNT/FST) word. An FNT/FST entry is the system's description of a file currently assigned to a job.

Frame

A tape recording unit made up of 1 bit from each tape track (7 bits for seven-track tape and 9 bits for nine-track tape). Each frame on a coded tape usually represents one character.

Generation

The position of a file within a series of files, each file developed from the preceding file. The generation number and generation version number of a tape file can be entered in its HDR1 label.

Graphic

A character that can be printed or displayed. Refer to definition of control character.

IAF

Interactive Facility. The network time-sharing application.

Indirect access file

A mass storage permanent file. Indirect access files can be accessed only through a working copy of the file. If requested, the working copy replaces the permanent file.

Input file type

Job file. Its first record is a control statement record which may be followed by records containing data, directives, or programs used by job steps.

Interrecord gap

Space skipped between the writing of data blocks on magnetic tape.

Job	A set of control statements and the data and directives used by those statements. A batch job must begin with the job and USER statements. A time-sharing job begins at the user's login to a terminal.
Job step	One of the sequence of operations performed within a job. Usually, each control statement results in one job step. However, a load sequence is a single job step that may result from several loader control statements.
Label	An 80-character block written to identify and/or delimit a tape volume or a file.
Level number	Octal number within a PRU terminator indicating the type of boundary it represents. A level 17 in an empty PRU is a NOS EOF; a level 0 in a short PRU is a NOS EOR; a level 1 in a short PRU is an EOR from an interactive terminal; and a level 16 in a short PRU is an EOF on a checkpoint file.
lfn	Name of a file assigned to the job (local file name).
Library	1. A collection of programs or routines. 2. A file containing records that are accessed individually. 3. A file searched by CYBER Loader for entry points referenced by a program. 4. A file containing compressed records in Modify or Update format.
Library file	A read-only file that can be accessed by several users simultaneously.
Line	A unit of data terminated by a zero-byte terminator. Unit used in time-sharing I/O, line printer output, and card reader input.
Load point	Metallic strip marking the beginning of the recordable portion of a magnetic tape.
Local file	1. A file type that refers to a temporary file other than the primary file. It often contains a copy of an indirect access file or data from a magnetic tape. 2. A file currently assigned to a job.
Logical record	A unit of data ending with an EOR.
Macro	A sequence of COMPASS source statements that can be called during assembly of a COMPASS program to produce code to perform a particular function.
Mass storage	Magnetic disk or extended memory.
Multifile file	A file containing more than one logical file. It begins at BOI and ends at EOI. On a labeled tape, a multifile file is delimited by corresponding HDR1 and EOF1 labels.

SAMPLE JOB OUTPUT

D

This appendix lists the output information printed for the sample job shown below. The notes in the right margin identify the various format conventions of NOS output. The job consists of the following statements.

```
TESTA.
USER(JEANCOM,PASSWRD)
CHARGE(JLC3951,27)
FTN.
/EOR
      PROGRAM CONVER(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C
C        INPUT -- 10-DIGIT OCTAL NUMBER, SUCH AS 0000000177
C        OUTPUT -- DECIMAL EQUIVALENT OF OCTAL NUMBER INPUT
C        IF AN 8 OR 9 DIGIT IS ENTERED, THE PROGRAM STOPS.
C        IF A NON-NUMERIC CHARACTER IS ENTERED, THE PROGRAM ABORTS.
C
      INTEGER NUM,DIGIT,SUM,PLVAL,PLACE
      DIMENSION NUM(10)
C
C     READS AN OCTAL NUMBER INTO THE NUM ARRAY
C
   10 READ(5,1000) NUM
 1000 FORMAT(10I1)
      IF (EOF(5)) 50,15
C
C     CHECKS IF AN 8 OR 9 WAS INPUT
C
   15 DO 20 I=1,10
   20 IF (NUM(I).EQ.8 .OR. NUM(I).EQ.9)   GO TO 50
C
C     CONVERTS THE OCTAL NUMBER BY MULTIPLYING EACH DIGIT
C     BY ITS PLACE VALUE AND ADDING IT TO A SUM
C
      DIGIT=10
      PLACE=0
      SUM=NUM(DIGIT)
   30 DIGIT=DIGIT-1
      PLACE=PLACE+1
      IF (DIGIT.EQ.0)    GO TO 40
      PLVAL=NUM(DIGIT) * 8 ** PLACE
      SUM=SUM+PLVAL
      GO TO 30
C
C     OUTPUTS CONVERTED NUMBER
C
   40 WRITE(6,4000) SUM
 4000 FORMAT(1X,1I10)
      GO TO 10
   50 STOP
      END
/EOR
/EOF
```

```
NOS 1                                    yy/mm/dd.
                        OPERATING SYSTEM
                     JOB ORIGIN = BATCH.

                     USER NUMBER = JEANCOM
                     JOBCARD NAME = TESTA00

 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
AAAAAAAAAAAAA  AAAAAAAAAAAAA  FFFFFFFF            II      AAAAAAAAAAAAA       JJ      CC
AAAAAAAAAAAAA  AAAAAAAAAAAAA  FF                  II      AAAAAAAAAAAAA       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  JJJJJJJ      CCCCCCCCCCCC
AA         AA  AA         AA  FF            IIIIIIIIIIII  AA        AA   JJJJJ        CCCCCCCCCC

                        yy/mm/dd.  hh.mm.ss.
```

The first three lines of the banner page indicate that this local batch job was printed under the control of the Network Operating System. The system creation date is specified by yy/mm/dd. (year/month/day.).

The user number is that user number supplied on the most recent USER statement before the job created the output file. The jobcard name is the name of the particular job which was supplied on the job statement.

The banner job name (AAFIAJC) is the same as the job's job name. Refer to Job Names in section 3 for an explanation of how job names are derived.

The last line specifies the current date (year/month/day.) and the time (hours.minutes.seconds.) when job printing was initiated.

Processing of the FTN. control statement compiles the source program in the second record of the job. The following is the program listing and symbolic reference map produced by that compilation.

```
      PROGRAM CONVER     73/74   OPT=1                          FTN 4.8+497        79/05/14. 09.55.13        PAGE     1

 1              PROGRAM CONVER(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
               C
               C       INPUT -- 10-DIGIT OCTAL NUMBER, SUCH AS 0000000177
               C       OUTPUT -- DECIMAL EQUIVALENT OF OCTAL NUMBER INPUT
 5             C       IF AN 8 OR 9 DIGIT IS ENTERED, THE PROGRAM STOPS.
               C       IF A NON-NUMERIC CHARACTER IS ENTERED, THE PROGRAM ABORTS.
               C
                       INTEGER NUM,DIGIT,SUM,PLVAL,PLACE
                       DIMENSION NUM(10)
10             C
               C       READS AN OCTAL NUMBER INTO THE NUM ARRAY
               C
                  10 READ(5,1000) NUM
                1000 FORMAT(10I1)
15                     IF (EOF(5)) 50,15
               C
               C       CHECKS IF AN 8 OR 9 WAS INPUT
               C
                  15 DO 20 I=1,10
20                20 IF (NUM(I).EQ.8 .OR. NUM(I).EQ.9)  GO TO 50
               C
               C       CONVERTS THE OCTAL NUMBER BY MULTIPLYING EACH DIGIT
               C       BY ITS PLACE VALUE AND ADDING IT TO A SUM
               C
25                     DIGIT=10
                       PLACE=0
                       SUM=NUM(DIGIT)
                  30 DIGIT=DIGIT-1
                       PLACE=PLACE+1
30                     IF (DIGIT.EQ.0)  GO TO 40
                       PLVAL=NUM(DIGIT) * 8 ** PLACE
                       SUM=SUM+PLVAL
                       GO TO 30
               C
35             C       OUTPUTS CONVERTED NUMBER
               C
                  40 WRITE(6,4000) SUM
                4000 FORMAT(1X,1I10)
                       GO TO 10
40                50 STOP
                       END


      SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
 4141  CONVER

VARIABLES     SN  TYPE           RELOCATION
 4220  DIGIT     INTEGER                        4224  I         INTEGER
 4225  NUM       INTEGER    ARRAY               4223  PLACE     INTEGER
 4222  PLVAL     INTEGER                        4221  SUM       INTEGER
FILE NAMES       MODE
    0  INPUT                2054  OUTPUT                0  TAPE5     FMT          2054  TAPE6     FMT

EXTERNALS          TYPE   ARGS
      EOF         REAL      1

STATEMENT LABELS
 4142  10                              0  15          INACTIVE              0  20
 4162  30                           4173  40                           4176  50
 4206  1000    FMT                  4214  4000    FMT

LOOPS  LABEL    INDEX     FROM-TO    LENGTH    PROPERTIES
 4150  20       I           19 20       7B     INSTACK    EXITS

STATISTICS
  PROGRAM LENGTH              353B     235
  BUFFER LENGTH              3664B    1972
        52000B CM USED
```

```
AJAICRR. 79/09/19. (0) CYBER 173 S/N 620 CLSH.

09.55.12.TESTA.
09.55.12.USER(JEANCOM,)
09.55.13.CHARGE(JLC3951,27)
09.55.13.FTN.
09.55.14.       .075 CP SECONDS COMPILATION TIME
09.55.14.UEAD,      0.002KUNS.
09.55.14.UEPF,      0.005KUNS.
09.55.14.UEMS,      0.703KUNS.
09.55.14.UECP,      0.082SECS.
09.55.14.AESR,      2.348UNTS.
09.55.21.UCLP, 6232,      0.192KLNS.
```

The first line specifies the job name, the current data; and the computer system and system version where the job executed. The dayfile includes a listing of the control statements, system-supplied status messages, and program output, if any. Spaces precede status messages and program output. Each line includes the time the message was issued to the dayfile.

The last six lines specify the type and amount of system resources the job used. This job used 0.002 kilounit of application activity, 0.005 kilounit of permanent file activity, 0.703 kilounit of mass storage activity, 0.082 second of central processor time, and 2.348 system resource units. The 6232 after UCLP gives the machine ID as 62 and the EST ordinal of the printer as 32. The job produced 0.192 kiloline (192 lines) or printable output.

Depending on the resources used, additional information may be included in the dayfile. Refer to Job Completion in section 3 for the formats of these messages.

TIME-SHARING INTERFACE E

The time-sharing interface, IAF or the time-sharing executive, processes communications between NOS and time-sharing terminals. When in the batch subsystem, the time-sharing user can enter any of the control statements described in this manual. He can also enter a number of control statements and commands intended for use only by time-sharing jobs. The time-sharing control statements and commands are described in the Network Products IAF Reference Manual and the NOS Time-Sharing User's Reference Manual.

Tables 1-A-1 and 1-A-2 list the time-sharing character sets. This appendix describes terminal character code conversion and the time-sharing control statements that can be included in procedure files.

TERMINAL CHARACTER CONVERSION

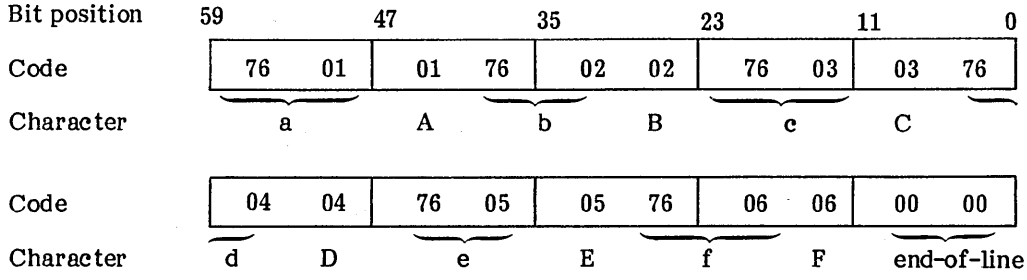
Normal input mode from an ASCII code terminal uses a 63- or 64-character set where all lowercase alphabetics are converted to uppercase characters. Under ASCII mode, the characters 74 and 76 represent the beginning of a 74xx or 76xx escape sequence. Under normal mode, the characters 74 and 76 are treated as data rather than escape codes. ASCII and normal modes apply to both input and output.

DATA INPUT

The manner in which the system interprets the characters entered from a terminal depends on whether the user requests ASCII or normal mode. For example, if the user enters

aAbBcCdDeEfF

when in ASCII mode, the central memory equivalent is:†



†Partial words are zero-filled; partial bytes are blank-filled.

However, if the user enters the characters in normal mode, the characters are mapped into the 64-character subset of the ASCII character set that contains only uppercase letters; then the central memory equivalent is:

Bit position	59		47		35		23		11	0
Code	01	01	02	02	03	03	04	04	05	05
Character	A	A	B	B	C	C	D	D	E	E
Code	06	06	00	00	00	00	00	00	00	00
Character	F	F	end-of-line							

Refer to appendix A for further description of the time-sharing code sets.

DATA OUTPUT

Data output is in either a 64/63- or 128-character set, depending on whether the terminal is in normal or ASCII mode. When the terminal is in normal mode, the codes 74 and 76 represent data rather than escape codes. In ASCII mode, 74 and 76 are treated as the beginning of an escape sequence.

For a more detailed description of terminal operation, refer to the Network Products IAF Reference Manual or the NOS Time-Sharing User's Reference Manual.

Data can also be transmitted to or from a terminal through a paper tape reader.

TIME-SHARING CONTROL STATEMENTS

The user can include the following control statements in a procedure file called from a time-sharing job in addition to using them as normal time-sharing commands. For more information on these statements, refer to the Network Products IAF Reference Manuals or the NOS Time-Sharing User's Reference Manual.

NOTE

If a time-sharing control statement is included in a nontime-sharing job, the system terminates the job.

ASCII STATEMENT

The ASCII control statement specifies that further terminal input and output is to be interpreted as 6/12 display code.

The control statement format is:

ASCII.

If this control statement is processed while output is still available, the terminal switches to ASCII mode for the remainder of the output.

CSET STATEMENT

The CSET control statement specifies the current code set of the terminal.

The control statement format is:

CSET(m)

m	Current terminal code set; m may be one of the following:	
	ASCII	Uses 6/12 display code set; escape code processing.
	NORMAL	Uses display code set; escape code processing is disabled.

If this control statement is processed while output is still available, the terminal switches to the new character set mode for the remainder of the output.

PARITY STATEMENT †

The PARITY control statement sets the terminal to the indicated parity.

The control statement format is:

PARITY(p)

p	Terminal parity; p may be one of the following:	
	ODD	Set odd parity.
	EVEN	Set even parity.

If p is omitted, odd parity is assumed.

If this control statement is processed while output is still available, the terminal parity switches to the new parity for the remainder of the output.

---

†Not available for IAF.

TRMDEF STATEMENT †

When executed by a time-sharing job, the TRMDEF control statement specifies changes in the characteristics of the network terminal that issued the statement. For a detailed description of this statement, refer to the Network Products IAF Reference Manual.

The control statement format is:

TRMDEF(L=lfn,xx_1=$value_1$,xx_2=$value_2$,...,xx_n=$value_n$)

L=lfn — Names optional local file to receive the terminal redefinition information. If it names a file other than the output file for the terminal, the terminal characteristics do not change until the file lfn is copied to the output file for the terminal.

xx_i=$value_i$ — Two-character parameter mnemonic equated to a legal value for that terminal characteristic. The legal value may be entered in one of the following formats.

y	Alphanumeric character.
y	Any character, including special characters.
yyyB	Octal value of an ASCII character.
Xyy	Hexadecimal value of an ASCII character.

The parameter mnemonics and the legal values for terminal characteristics are listed in the Network Products IAF Reference Manual.

---

†Valid only from IAF.

CARD FILE DATA CONVERSION F

Data within the system is stored in binary or coded format. Binary data is variable length central memory images. Coded data consists of display-coded characters. Each coded line is stored as an even number of characters. If an odd number of characters is entered, the system appends a space to make it even.

This appendix describes the formats for punch cards. It also describes the conversion performed by the system on data transferred between the system and card readers and punches.

When using the 64-character set, the user should avoid using consecutive colons (00 characters). It is possible for these colons to be interpreted as an end-of-line. An end-of-line is defined as 12 to 66 zero bits, right-justified in one or two central memory words. If consecutive colons appear in the lower 12 bits of a central memory word, they are interpreted as an end-of-line rather than as colons.

Example:

The following characters are punched on a coded card beginning in column 1.

::::::::A::::::::AA

This appears in memory as follows:

Bit position	59		47		35		23		11	0
Code	00	00	00	00	00	00	00	00	00	01
Character	:	:	:	:	:	:	:	:	:	A
Code	00	00	00	00	00	00	00	00	01	01
Character	:	:	:	:	:	:	:	:	A	A
Code	00	00	00	00	00	00	00	00	00	00
Character	end-of-line									

However, if the characters were copied with the COPYSBF utility, the following appears.

Bit position	59		47		35		23		11	0
Code	55	00	00	00	00	00	00	00	00	00
Character						end-of-line				
Code	01	00	00	00	00	00	00	00	00	01
Character	A	:	:	:	:	:	:	:	:	A
Code	01	55	00	00	00	00	00	00	00	00
Character	A					end-of-line				

Because the COPYSBF utility shifts each line one 6-bit character to the right and adds a space, copying nine colons puts 12 zero bits in the last byte of the first word. This is interpreted as an end-of-line.

NOTE

If a colon is the last character of an input line, the system appends a space to preserve the colon and then appends an end-of-line. If needed, a second space is added to ensure an even number of 6-bit characters. Refer to figure 1-F-1.

INPUT CARD FILE FORMATS

The system reads cards in coded and binary formats. The following conditions apply in both formats.

- A card with a 7/8/9 punched in column 1 is an EOR mark.
- A card with a 6/7/9 punched in column 1 is an EOF† mark.
- A card with a 6/7/8/9 punched in column 1 is an EOI mark.

The remainder of each card is ignored except for columns 79 and 80 of the EOR and EOF cards. These columns can contain the keypunch conversion mode for the input records that follow.†† Conversion modes are discussed in Coded Cards.

† The 6/7/9 keypunch mark is not supported by either Export/Import or RBF.
†† HASP terminals can support other forms of separator cards such as /*EOR and /*EOI. (Refer to the Network Products RBF Reference Manual.)

CODED CARDS

Cards are read in Hollerith punch code. The 3447 card reader controller converts the Hollerith code to internal BCD code and passes the data to the card reader driver. The driver converts the data from internal BCD code to display code. Up to 80 characters can be transferred per card. Trailing blank bytes are deleted. If a line has an odd number of characters, one trailing blank is added to make it even. In order to preserve the colon (00_8) of the 64-character set, a trailing blank byte is either retained or appended as the last character in an even line. Examples of coded card conversion are shown in figure 1-F-1.

Conversion Modes

Two conversion modes, O26 and O29,† exist for the Hollerith punch code. All data is converted in the system default keypunch mode unless a conversion mode change is specified. This change can be specified on any of the following cards.

The job card, 7/8/9 card (EOR mark), and 6/7/9 (EOF mark) can contain the keypunch conversion mode in columns 79 and 80. A 26 punched in columns 79 and 80 indicates that all subsequent cards are converted in O26 mode. A 29 indicates that subsequent cards are converted to O29 mode. Each conversion change remains in effect until another change card is encountered or the job ends. The user can switch between O26 and O29 mode as often as desired. If 26 or 29 does not appear in columns 79 and 80 of the job card, the initial keypunch mode of that job is the system default mode. If 26 or 29 does not appear on a 7/8/9 or 6/7/9 card, no conversion change is made, and the most recent keypunch mode remains in effect.

Keypunch mode can also be changed by a card containing a 5/7/9 punch in column 1. A blank (no punch) in column 2 indicates O26 conversion mode; a 9 punch in column 2 indicates O29 mode. The conversion change remains in effect until another change card is encountered or the job ends.

Literal Input

The 5/7/9 card also allows literal input when 4/5/6/7/8/9 is punched in column 2. Literal input allows 80-column binary data to be read while transmitting input in coded mode. Cards are read (16 central memory words per card) until a card identical to the previous 5/7/9 card (4/5/6/7/8/9 in column 2) is read. The next card can then specify the new conversion mode.

In order to maintain system integrity, an end-of-information card always terminates 80-column binary input (literal input). Either of the following is interpreted as an end-of-information card even though it appears in a literal input record.

- A card with 6/7/8/9 punched in column 1 and with columns 2 through 80 blank.
- A card with 6/7/8/9 punched in columns 1 and 80 and with columns 2 through 39 and columns 41 through 79 blank. Column 40 may be punched or left blank.

† These codes are ignored by a 200 User Terminal since conversion mode is selected by a hardware switch. (Refer to the NOS Export/Import Reference Manual and the Network Products Remote Batch Facility Reference Manual.)

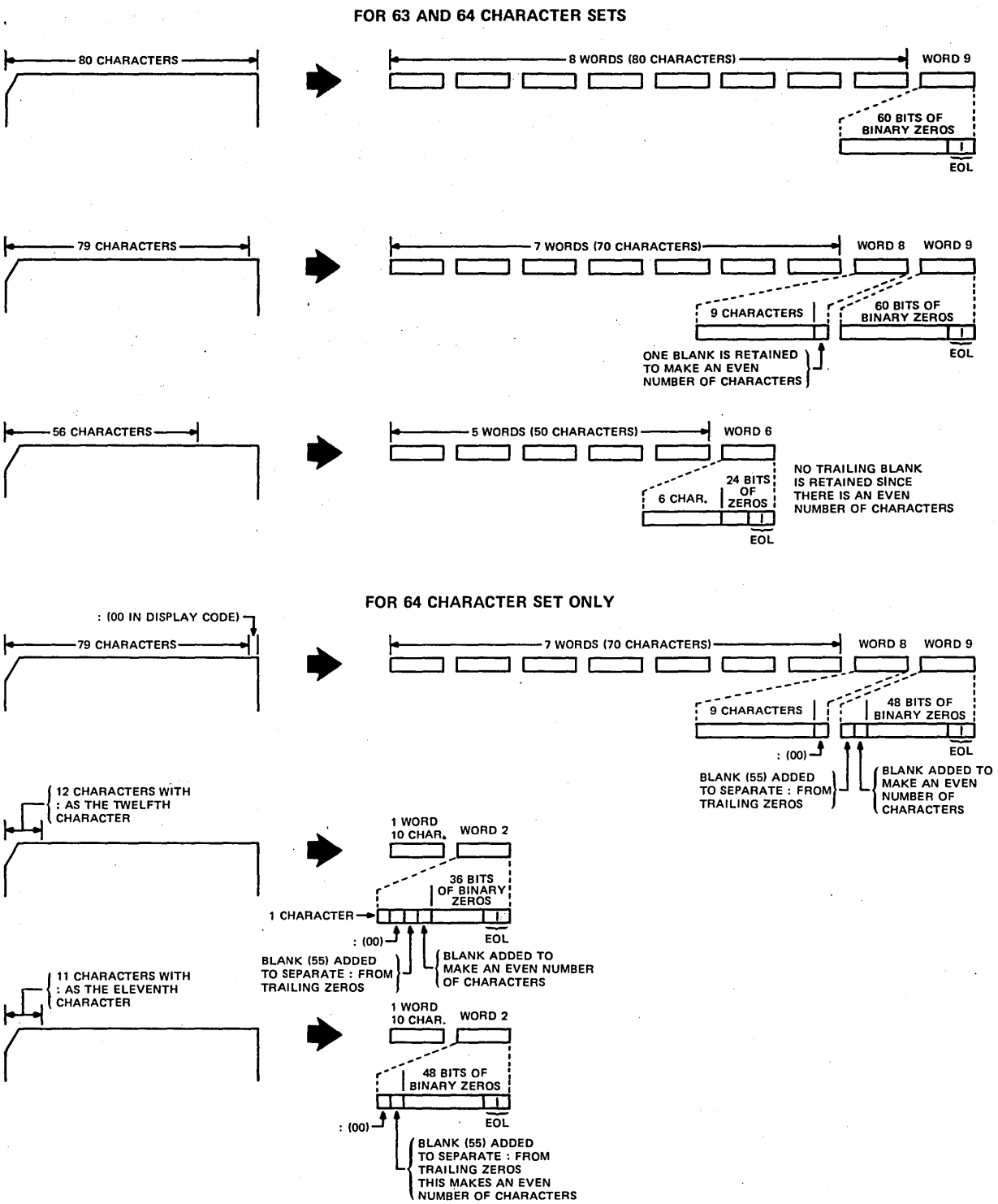


Figure 1-F-1. Examples of Coded Card Conversion

BINARY CARDS

Binary cards are denoted by a 7/9 punch in column 1 and can contain up to 15 central memory words. The 3447 card reader controller reads the binary data and passes it to the card reader driver in 12-bit codes. Each card column row corresponds to a bit position. The driver checks the checksum figure if this option is specified. The driver then passes the data to the central memory buffer.

The fields within a binary card are:

Column(s)	Description
1	7/9 punch indicates a binary card. 4 punch ignores checksum punch in column 2. Rows 0, 1, 2, and 3 contain the binary equivalent of the word count of the card.
2	Binary data checksum (modulo 4095).
3 through 77	Fifteen central memory words of binary data.
78	Blank.
79 and 80	24-bit binary card sequence number.

SUMMARY

The following punches appearing in column 1 of a card have the corresponding meaning to the card reader driver.

Punch	Represents
7/8/9	End-of-record (optional conversion mode change).
6/7/9	End-of-file (optional conversion mode change).
6/7/8/9	End-of-information.
5/7/9	Conversion mode change/read 80-column binary.
7/9	Binary card.
Not 7 and 9	Coded card.

PUNCH FILE FORMATS

Punched cards can be in three formats.

- Coded (Hollerith punch).
- Binary.
- Absolute binary.

The following conditions apply to all three formats.

- When an EOR is encountered, a card is punched with a 7/8/9 in columns 1 and 80. This card is offset.
- When an EOF is encountered for a file, a card is punched with a 6/7/9 in columns 1 and 80; the remainder of the card is blank. This card is offset.
- When an EOI is encountered on a file, a card is punched with a 6/7/8/9 in columns 1 and 80; the remainder of the card is blank. This card is offset.
- If a compare error is encountered, the erroneous card and the following card are offset. These two cards are repunched until no error is detected. An EOI card with 6/7/8/9 punches in columns 1 and 80 contains a binary count in column 40 of the number of compare errors.
- During the punching of each file, the system maintains a count of the number of cards punched for the file. If the number exceeds the limit for which the user is validated, punching of the file is terminated. A special banner card with the word LIMIT is punched and offset as the last card of the deck.

The following methods are used by the system to punch each of the three forms of cards.

CODED CARDS (PUNCH)

With the exception of decks punched via the ROUTE or 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	Fifteen central memory words of binary data.
78	Blank.
79 and 80	24-bit binary card sequence number.

ABSOLUTE BINARY CARDS (P8)

Absolute binary cards are central memory images in 12-bit codes. Each row in a card column corresponds to a bit position. Sixteen central memory words are punched per card with no special punches or fields added.

ANSI TAPE LABEL FORMATS G

ANSI labels perform two functions. They provide information that uniquely identifies a file and the reel on which it resides, and they mark the BOI and EOI of a file and the beginning and end of a reel.

ANSI labels are designed to conform to the American National Standard Magnetic Tape Labels for Information Interchange X3.27-1969. All labels are 80 characters long and are recorded at the same density as the data on the tape. The first 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 As	Required/Optional
VOL	1	Volume header label	Beginning-of-volume	Required
UVL	1-9	User volume label	Beginning-of-volume	Optional
HDR	1	File header label	Beginning-of-information	Required
HDR	2-9	File header label	Beginning-of-information	Optional
UHL		User header label	Beginning-of-information	Optional
EOF	1	End-of-file label	End-of-information	Required
EOF	2-9	End-of-file label	End-of-information	Optional
UTL	†	User trailer label	End-of-information	Optional
EOV	1	End-of-volume label	End-of-volume	Required when appropriate
EOV	2-9	End-of-volume label	End-of-volume	Optional

REQUIRED LABELS

The VOL1, HDR1, and EOF1 labels are required on all ANSI-labeled tapes. In addition, an EOV1 label is required if the physical end-of-tape reflector is encountered before an EOF1 label is written or if a multifile set is continued on another volume. In the descriptions of the contents of these labels, n is any numeric digit and a is any letter, digit, or any of the following special characters.

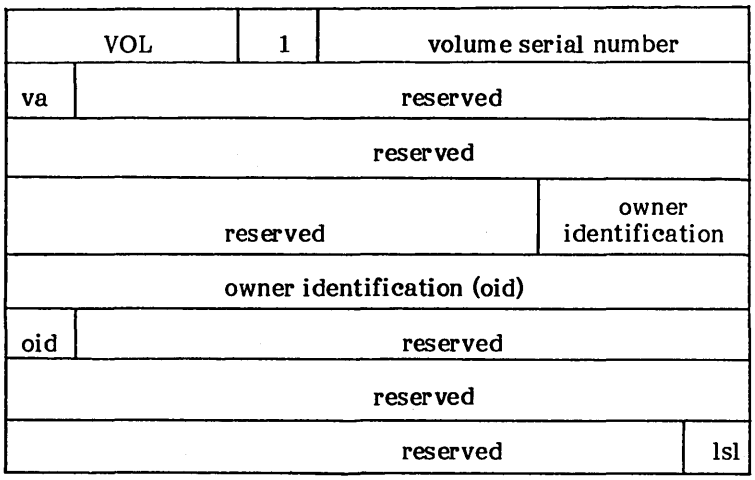
† Any member of the CDC 6-bit subset of the ASCII character set.

Δ	)	<
!	*	=
"	+	>
≠	,	?
$	-	@
%	.	[
&	'	\
/	:	]
(	;	^

Some fields are optional. An optional field which does not contain the designated information must contain blanks. Fields which are not described as optional are required and written as specified. n-type fields are right-justified and zero-filled, and a-type fields are left-justified and blank-filled.

VOL1 — VOLUME HEADER LABEL

The volume header label must be the first label on a labeled tape. All reels begin with a VOL1 label. If two or more reels belong to a volume set, the file section field in the following HDR1 label gives the actual reel number.



Character Position	Field Name	Length (in Characters)	Contents	Default	Checked on Read	Checked on Overwrite
1-3	Label identifier	3	Must be VOL.		Yes	No
4	Label number	1	Must be 1.		Yes	No
5-10	Volume serial number	6	Volume identification assigned by owner to identify this physical reel of tape. If the volume serial number is all blanks, the tape is a scratch tape.	As read from existing label	Yes, if the file was assigned by volume serial number.	No
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).	Yes
12-31	Reserved for future standardization	20	Must be blanks.		No	No
32-37	Reserved for future standardization	6	Must be blanks.		No	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.	Yes
52-79	Reserved for future standardization	28	Must be blanks.		No	No
80	Label standard level (lsl)	1	A 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	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="8">file identifier (fi)</td></tr>
<tr><td>fi</td><td colspan="4">set identification</td><td colspan="3">file section number (secno)</td></tr>
<tr><td>secno</td><td colspan="2">file sequence number</td><td colspan="4">generation number</td><td>gvn</td></tr>
<tr><td>gvn</td><td colspan="4">creation date</td><td colspan="3">expiration date</td></tr>
<tr><td colspan="3">expiration date</td><td>fa</td><td colspan="4">block count</td></tr>
<tr><td colspan="8">system code</td></tr>
<tr><td colspan="4">system code</td><td colspan="4">reserved</td></tr>
</table>

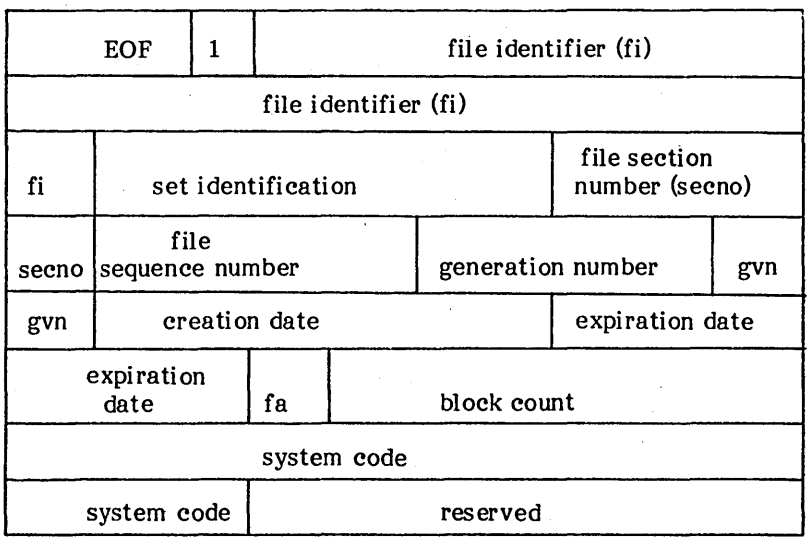
Character Position	Field Name	Length (in Characters)	Contents	Default	Checked on Read	Checked on Overwrite
1-3	Label identifier	3	Must be HDR.		Yes	No
4	Label number	1	Must be 1.		Yes	No
5-21	File identifier (fi)	17	Up to 17 a characters used as the file identification (fileid) parameter on the LABEL control statement.	Blank	Checked if specified.	No
22-27	Set identification	6	Up to six a characters used as the setid parameter on the LABEL control statement. To conform to the ANSI tape standard, this value is the same for all files of a multifile set.	Blank; an appended file is given the same set identification as its preceding file.	Checked if specified.	No
28-31	File section number (secno)	4	Four n characters identifying the file section number. The file section number of the first HDR1 label of a file is 0001. If the file extends to more than one volume, this number is incremented by one for each subsequent volume. This value corresponds to the secno parameter on the LABEL statement.	0001	Checked if specified.	No
32-35	File sequence number	4	Four n characters used as the seqno parameter on the LABEL statement. This parameter specifies the position of a file within a file set. This value is 0001 for the first file, 0002 for the second, and so on. In all the labels for a given file, this field contains the same number.	0001	Checked if specified.	No

Character Position	Field Name	Length (in Characters)	Contents	Default	Checked on Read	Checked on Overwrite
36-39	Generation number (optional)	4	Four n characters specifying the generation number of a file. This is the genno parameter of the LABEL statement. This value is 0001 for the first generation of a file, 0002 for the second, and so on.	0001	Checked if specified.	No
40-41	Generation version number (grn)	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	No
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.	No
48-53	Expiration date	6	The file is considered expired when today's date is the same as 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 earlier than or the same as the expiration date of	Current date	No	Yes

Character Position	Field Name	Length (in Characters)	Contents	Default	Checked on Read	Checked on Overwrite
			all preceding files on the volume. The expiration date is written in the same format as the creation date. 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.	Yes
55-60	Block count	6	Must be zeros.		No	No
61-73	System code	13	Thirteen 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	No
74-80	Reserved for future standardization	7	Must be spaces.		No	No

EOF1 — FIRST END-OF-FILE LABEL

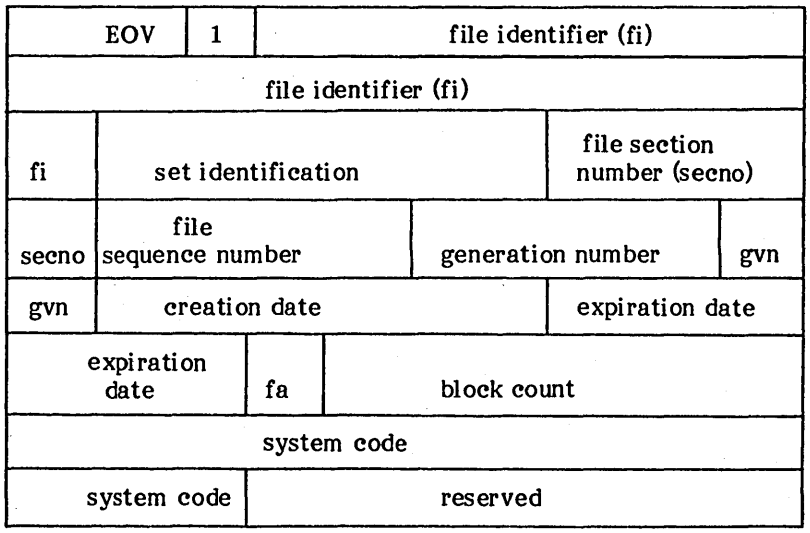
The end-of-file label is the last block of every file. It is the system end-of-information for the file. A single tape mark precedes EOF1. A double tape mark written after the EOF1 label marks the end of a multifile set.



Character Position	Field Name	Length (in Characters)	Contents	Default	Checked on Read
1-3	Label identifier	3	Must be EOF.		Yes
4	Label number	1	Must be 1.		Yes
5-54	Same as corresponding fields in HDR1 (optional)	50	Same as the corresponding fields in HDR1.		Same as HDR1.
55-60	Block count	6	Six n characters specifying the number of data blocks between this label and the preceding HDR label group. This total does not include labels or tape marks.		Yes
61-80	Same as corresponding fields in HDR1 (optional)	20	Same as corresponding fields in HDR1.		Same as HDR1.

EOV1 — FIRST END-OF-VOLUME LABEL

The end-of-volume label is required only if the physical end-of-tape reflector is encountered before an EOF1 label is written or if a multifile set is continued on another volume. EOV1 is preceded by a single tape mark and followed by a double tape mark.



Character Position	Field Name	Length (in Characters)	Contents	Default	Checked on Read
1-3	Label identifier	3	Must be EOV.		Yes
4	Label number	1	Must be 1.		Yes
5-54	Same as 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.

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), user volume (UVLa), header (UHLa), and trailer (UTLa) labels.

HDR2 THROUGH HDR9 — ADDITIONAL FILE HEADER LABELS

HDR2 through HDR9 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 THROUGH EOF9 — ADDITIONAL END-OF-FILE LABELS

EOF2 through EOF9 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 THROUGH EOV9 — ADDITIONAL END-OF-VOLUME LABELS

EOV2 through EOV9 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		

Refer to section 3 in volume 2 for a description of the use of EOV2 labels in conjunction with CLOSER, REWIND, and UNLOAD macros.

USER LABELS

User labels may immediately follow their associated system labels. Thus, user volume labels (UVLa) may follow VOL1, user header labels (UHLa) may follow the last HDRn label, and user trailer labels (UTLa) may follow the last EOVn or EOFn label. Their format is:

Character Position	Field Name	Length (in Characters)	Contents	Default Written
1-3	Label identifier	3	UVL, UHL, or UTL.	UVL, UHL, or UTL.
4	Label number	1	Must be 1, 2, 3, 4, and so on, consecutively for UVL labels. For other labels, any a character.	
5-80	User option	76	Any a characters.	

Only the label identifier and the label number are checked on read. The system checks the number of user labels of a label type; a maximum of 64 is allowed.

CONTROL LANGUAGE (KCL) H

NOTE

The control language (KCL) described in this appendix is the system control language available under NOS prior to the introduction of the CYBER Control Language (section 4). Support of KCL will be dropped in a future NOS release, so users are encouraged to convert their KCL procedures to CYBER Control Language (CCL) and not to mix the KCL and CCL statements. The CCL BEGIN statement mixed with the KCL GOTO or CALL statements may give unpredictable results.

The following paragraphs describe the various components and statements of KCL.

EXPRESSIONS

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

OPERATORS

The arithmetic, relational, and logical operators are the same in the control language outlined in this appendix as they are in the CYBER Control Language described in section 4. The only exception is the exclusive OR logical operator (.EOR.).

FUNCTIONS

Two functions are provided for use in expressions specified with control language statements. The FILE function determines the status of any file assigned to the job. The NUM function determines if a specified parameter name has a numeric value. For complete information concerning format and use, refer to Control Language Functions in this section.

SYMBOLIC NAMES

Symbolic names are used to reference values pertaining to the job process. There are three categories of symbolic names, as follows:

- Symbolic names with fixed arithmetic values:

ARE	Arithmetic error.
BCO	Local batch origin.
CMM	Maximum CM field length (MFL setting).
CMN	Nominal CM field length (RFL setting).
CPE	CPU abort.
ECM	Maximum ECS field length.
ECN	Nominal ECS field length.
EIO	Remote batch origin.
FLE	File limit error.
FSE	Forced error.
MNE	Monitor call error.
ODE	Operator drop.
OKE	Operator kill drop.
PPE	PPU abort.
PSE	Program stop error.
RRE	Rerun error.
SRE	SRU limit error.
SSE	Subsystem aborted.
SYO	System origin.
TKE	Track limit error.
TLE	Time limit error.
TXO	Time-sharing origin.

- Symbolic names with variable arithmetic values which depend upon job state:

EF	Previous error flag.
EM	Current exit mode.
FL	Job field length.
OT	Job origin type.
R1	Contents of control register 1.
R2	Contents of control register 2.
R3	Contents of control register 3.
SS	Job subsystem. This particular symbolic name requires an equal sign. SS may be equivalenced to one of the following:

ACCESS

BASIC

BATCH

EXECUTE

FORTRAN

FTNTS

NULL

TRANACT (for TAF/TS only)†

- Symbolic names with Boolean values:

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

EVALUATION OF EXPRESSIONS

The order of evaluation of expressions is:

1. Exponentiation
2. Multiplication, division

†Special validation is necessary to use the library update and batch transaction functions of TAF/TS. Refer to the LIMITS statement in section 6.

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. The C and S options are order independent; the RENAME option, if present, must be last.

Lines within a procedure file may contain line numbers to make maintenance easier. Usually, the CALL statement strips off these line numbers before copying the procedure statements to the new control statement record. However, if a comma immediately follows the line number, the line number remains on the statement.

The statement format is:

CALL(lfn,C,S=ccc,RENAME(oldnam$_1$=newnam$_1$,oldnam$_2$=newnam$_2$,...,oldnam$_n$=newnam$_n$))

or

CALL(lfn,C,S=ccc(oldnam$_1$=newnam$_1$,oldnam$_2$=newnam$_2$,...,oldnam$_n$=newnam$_n$))

lfn — Procedure file name (refer to the description of procedure files in this section for further information). The system obtains lfn by:

- Searching for a local file, lfn.
- Searching the system library for lfn.
- Attempting to retrieve a working copy of an indirect access file.

C — Replaces all of the control statement record after the CALL statement with lfn.

S=ccc — Sets next control statement to be processed to statement ccc. If S is not specified, the first statement in lfn is processed.

RENAME — Each occurrence of oldnam$_i$ is replaced with newnam$_i$ before the statement is entered into the statement file. As shown by the optional format, the word RENAME does not have to appear.

oldnam$_i$ — Old name; name of a file or statement label used in the specified procedure file.

newnam$_i$ — New name; name to replace oldnam$_i$.

If lfn is not properly formatted for a procedure file, the following message is issued.

lfn NOT A PROCEDURE FILE.

DISPLAY STATEMENT

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

The format of the DISPLAY statement is:

DISPLAY(exp)

exp A KCL expression.

Example:

The following sample dayfile shows several display operations.

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

The first DISPLAY statement displays the value of the TIME symbolic name. The current time given is in the form hhmm. The next six lines demonstrate the use of the R1 and R2 symbolic names. The other DISPLAY statements specify numeric expressions.

SET STATEMENT

The SET statement assigns a value to a control register, an error flag, or the flag that determines whether skipped control statements are entered in the dayfile. Using the SS function, it also can change the current time-sharing subsystem.

To assign a value to a symbolic name, the following format is used.

SET(sym=exp)

sym One of the following symbolic names (initially these names are set to 0).

Name	Description
R1, R2, or R3	Local control registers. When a procedure is called, the current values of R1, R2, and R3 are passed to the procedure. The values of these registers may change within the procedure; however, when processing reverts, these registers are restored to the values they had when the procedure was called.
EF	Local error flag. When a procedure is called, the current value of the error flag is passed to the procedure. The value of the error flag may change within the procedure; however, when processing reverts, the error flag is restored to the value it had when the procedure was called.

exp A KCL expression. The value derived through evaluation of the expression is assigned to the symbolic name. Acceptable values for each symbolic name follow.

sym	Suggested Value
R1, R2, or R3	Any integer between -131 071 and 131 071. If the value is outside this range, it is truncated. KCL does not issue a message as a result of the truncation.
EF	Any integer between 0 and 63. If the value is greater than 63, it is truncated. To assign the value defined by the system for an error condition, the user should set the error flag to one of the error condition symbolic names (refer to Symbolic Names at the beginning of this section). KCL sets the EF flag to the appropriate error code when an error occurs.

To change the current time-sharing subsystem, the following format is used.

SET(SS=subsystem)

subsystem Subsystem name. The subsystem names are ACCESS, BASIC, BATCH, EXECUTE, FORTRAN, FTNTS, NULL, and TRANACT.†

†TRANACT is not applicable to IAF.

Example:

This example illustrates the use of the SET statement to control execution of an object program. Because register R1 is set to 1 when the file ABC is called, TAPE 1 is not requested and the object program is not executed.

```
SET(R1=1)
CALL(ABC)
FTN.
IF(R1=1) GOTO,3.
REQUEST(TAPE1)
LGO.
3,REWIND(TAPE1)
    .
    .
    .
```

IF STATEMENT

The IF statement is used to evaluate an expression. If the conditions given in the expression are true, the dependent statement is processed. The expression is considered true if it is evaluated to a nonzero numeric value.

The statement format is:

IF(expression)stmt.

or

IF(SS=ssname)stmt.

expression	Any legal expression.†
stmt	Any legal control statement.
ssname	Any legal SS subsystem name.

NOTE

A statement of the form IF(expression)CALL (lfn) is not recommended. Each time the IF statement is processed and the expression is true, the CALL statement is processed. This merges the called lfn with the current control statement stream and creates a copy of this procedure file each time.

† If a permanent file control statement is included in an IF statement, a password (if present) is not deleted in the dayfile.

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=BASIC)GOTO,100.
SET(SS=BASIC)
    .
    .
    .
100,OLD,BAS.
```

If the statement is true, the OLD control statement is processed; otherwise, the SET statement is processed.

CONTROL LANGUAGE FUNCTIONS

Control language functions are described in the following paragraphs. Separators and terminators must be used as shown in the function formats.

FILE FUNCTION

The FILE function is used to determine the status of any file assigned to the job and is used in conjunction with the SET, IF, and DISPLAY control language statements.

The format of the function is:

FILE(lfn,expression)

lfn — File name.

expression — Any legal expression; however, FILE expressions cannot include functions. In addition, FILE expressions use different symbolic names, as follows:

Symbolic names:

Names with values:

EQ — Equipment status table (EST) ordinal† (0 through 77_8).

ID — File ID (0 through 67_8).

†Contact installation personnel for a list of EST ordinals.

Symbolic names:

File characteristics:

MS	File is on mass storage.
LK	File is locked.
OP	File is opened.
EX	Execute-only file.
AS	File is assigned to user's control point.

File types:

LO	Local.
PR	Print.
IN	Input.
PH	Punch.
LI	Library.
PM	Direct access permanent file.
PT	Primary.

Device types:

CP	415 Card Punch.
CR	405 Card Reader.
DE	Extended core storage.
DI	844-21 Disk Storage Subsystem (half track).
DJ	844-41 or 844-44 Disk Storage Subsystem (half track).
DK	844-21 Disk Storage Subsystem (full track).
DL	844-41 or 844-44 Disk Storage Subsystem (full track).
DM	885 Disk Storage Subsystem (half track).
DP	Distributive data path to ECS.
DQ	885 Disk Storage Subsystem (full track).
LP	Any line printer.
LR	580-12 Line Printer.
LS	580-16 Line Printer.
LT	580-20 Line Printer.

MS	Mass storage.
MT	Magnetic tape drive (seven-track).
NE	Null equipment.
NP	Host communications processor.
NT	Magnetic tape drive (nine-track).
TT	Time-sharing terminals.

Examples:

```
SET(R1=FILE(TAPE,MT))
```

If TAPE is a file on a seven-track magnetic tape drive, R1 is set to 1; otherwise, it is set to 0.

```
IF(FILE(BETA,DI.AND.PM))GOTO,200.
```

If BETA is a file on an 844-21 Disk Storage Subsystem and it is a direct access permanent file, control skips to the statement at 200.

NUM FUNCTION

The NUM function is used to determine if the specified parameter name has a numeric value. It is used in conjunction with the SET, IF, and DISPLAY control language statements.

The format of the function is:

```
NUM(name)
```

name	Parameter name. If the name is numeric, the statement is true; otherwise, it is false.

Example:

If the CALL statement

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

is used to call procedure file A, the IF statement in A

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

is evaluated as true, and control transfers to 1S.

However, the statement

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

is evaluated as false, and control passes to the next statement in A.

PROCEDURE FILES

Procedure files are source files consisting of control statements, control language statements, or both. The first statement of a procedure file may be the file name. If the first statement is the same as the file name used in the CALL statement, the first statement is ignored. Procedure files are activated by the CALL statement or by using the name of the procedure file, if the file is in the system.

Example 1:

The procedure file in this example is an indirect access file called COMPARE. This routine copies an input file and compares it with an existing direct access file. In the procedure file, these two files are called DUPL and MASTER. When the procedure file is inserted into the control statement record during job processing, the name of DUPL is changed to NEWFILE.

Original Input File

```
JOBAAA.
USER(EFD2501,PASS)
CHARGE(59,69N1)
CALL(COMPARE(DUPL=NEWFILE))
-EOR-
    .
    input file
    that is to
    be compared
    .
-EOI-
```

Procedure File COMPARE

```
COMPARE
COPYBR(,DUPL)
ATTACH(MASTER)
VFYLIB(MASTER,DUPL)
```

After the CALL control statement is processed, the control statement record is as follows:

```
JOBAAA.
USER(EFD2501,PASS)
CHARGE(59,69N1)
CALL(COMPARE(DUPL=NEWFILE))
COPYBR(,NEWFILE)
ATTACH(MASTER)
VFYLIB(MASTER,NEWFILE)
-EOR-
```

Example 2:

This is an example of nested calls. It illustrates the use of one procedure file to skip a specified number of files on a tape (contents of R1) and to copy source data to the tape. The other procedure file retrieves source data from the OPL (old program library) and calls the first procedure file to place that source data on the tape.

Input Deck

```
JOBAAA.
USER(USERNUM,PASSWRD,FAM1)
CHARGE(59,69N1)
ATTACH(OPL/UN=LIBRARY)
REQUEST(TAPE)
MODIFY(S,Z)/*EDIT,CPM
SET(R1=0)
CALL(PROC,RENAME(A=TAPE,B=SOURCE,2=2A,3=3A)
SET(R1=R1+1)
CALL(PROB)
-EOR-
```

Procedure File PROB

```
PROB
MODIFY(S=NEW,Z)/*EDIT,MTR
CALL(PROC,RENAME(A=TAPE,B=NEW)
RETURN,NEW.
```

Procedure File PROC

```
PROC
REWIND(A,B)
SET(R2=0)
2,IF(R1=R2)GOTO,3.
SKIPF(A)
SET(R2=R2+1)
GOTO,2.
3,COPYBF,B,A.
```

NOTE

On job initiation, the user's input file is a locked file. If the user wishes to call procedure files that write data on the input file, he should enter the RETURN(INPUT) control statement before attempting to write on INPUT. For further information, refer to Input File Control in section 3.

LINE PRINTER CARRIAGE CONTROL I

This appendix briefly describes the format and processing of print files.† It lists the carriage control for the programmable format control (PFC) and non-PFC 580 line printers.

PRINTED DATA

All data to be printed is in coded format in a print file within the print queue. The data consists of either 6-bit or 12-bit codes. Data recorded using the 6/12 display code set (refer to appendix A) should be converted to the 12-bit ASCII code set (refer to the FCOPY statement in section 7) before being routed to a line printer.

The system extracts data until an end-of-line occurs or until 137 characters are retrieved. End-of-line is 12 or more zero bits in the rightmost byte of a central memory word.

CARRIAGE CONTROL

The system interprets the first character in a line as the carriage control character†† and that character is not printed (table 1-I-1). The remainder of the line is then printed, except when the Q, R, S, or T carriage control characters are specified. The Q, R, S, T, and V format controls remain in effect until changed; all other carriage control characters must be supplied for each line they control. Line spacing is normally done in auto eject mode; that is, creases in the paper are skipped by the line printer's automatic line spacing mechanism if the paper is loaded properly. Auto eject mode must be turned off if the user wants to select format channels to advance printing from a position above the bottom of form to a position beyond the next top of form.

During the printing of each file, the system maintains a count of the number of lines printed or skipped for the file. If the number exceeds the limit for which the user is validated, printing of the file is terminated. The informative diagnostic LINE LIMIT EXCEEDED is printed. If a job's dayfile is part of the terminated print file, the dayfile is subsequently printed.

The installation can impose an implied page control by setting a certain number of default lines for each page. If less than the default number of lines is printed or skipped on a page, the line limit is still decremented by the default number of lines.

† To print a file in which the first character of each line is not a carriage control character, refer to the COPYSBF control statement in section 7.

†† The information in this appendix does not apply to remote batch line printers.

TABLE 1-I-1. CARRIAGE CONTROL CHARACTERS

Character	Action
SPACE	Single space.
1	Eject page before print.
0	Skip one line before print (double space).
-	Skip two lines before print (triple space).
+	Suppress space before print.
/	Suppress space after print.
2	Skip to last line of form before print.
Q	Clear auto eject; † remainder of line is not printed.
R	Set auto eject; remainder of line is not printed.
S	Select 6 lines/inch; remainder of line is not printed.
T	Select 8 lines/inch; remainder of line is not printed.
V	Eject page before print on a 580 PFC printer, V loads a user-supplied PFC array (validated users only).
8	Skip to next punch in format channel 1 before print. ††
7	Skip to next punch in format channel 2 before print. ††
6	Skip to next punch in format channel 3 before print. ††
5	Skip to next punch in format channel 4 before print. ††
4	Skip to next punch in format channel 5 before print. ††
3	Skip to next punch in format channel 6 before print. ††
H	Skip to next punch in format channel 1 after print.
G	Skip to next punch in format channel 2 after print.
F	Skip to next punch in format channel 3 after print.
E	Skip to next punch in format channel 4 after print.
D	Skip to next punch in format channel 5 after print.
C	Skip to next punch in format channel 6 after print.

† The deselection of auto eject mode on a 580 line printer results in the deselection of 8 lines per inch, if previously selected.
†† No space after print. For all other control characters, a line feed is issued after print.

CARRIAGE CONTROL USING FORMAT CHANNEL SELECTION

A programmer can use carriage control characters (table 1-I-1) that refer to channels (tracks) on a punched carriage control tape or in a programmable format control (PFC) array. The carriage control character beginning a print file line determines where on the printer form the line is printed.

After reading the carriage control character for a print line, the printer checks the format channel the character references. If a punch exists at that frame in that channel of the carriage tape or if a bit is set in that frame and channel of the PFC array, the printer prints the line. If not, the printer advances the carriage tape or PFC array and advances the printer paper until a punch or set bit is found in that channel. It then prints the line.

As listed in table 1-I-1, some carriage control characters name an action to be taken while others directly name a format channel. When specifying characters that name an action, the user indirectly names the format channel that performs the named action. To use the characters that directly name a format channel, the user needs to know the contents of the channel.

The format of the carriage control tape recommended for use on a 580 non-PFC line printer is listed in table 1-I-2. Lines 132 through 134 are identical to lines 0 through 2 because they overlap when the punched tape is glued together to form a continuous loop. Therefore, the user can disregard lines 132 through 134.

Selecting format channels on the carriage tape illustrated in figure 1-I-1 produces the following actions.

Carriage Control Character	Format Channel Selected	The line printer advances to:
8 or H	1	Line 0 or 66.
7 or G	2	First line of the next two-line group.
6 or F	3	First line of the next three-line group.
5 or E	4	First line of the next four-line group.
4 or D	5	First line of the next five-line group.
3 or C	6	Line 0.

If the numeric carriage control character is specified, the printer advances before printing. If the alphabetic character is specified, the printer advances after printing.

Frame	Channels											
	1	2	3	4	5	6	7	8	9	10	11	12
0	x	x	x	x	x	x	x	x	x	x	x	
1											x	
2		x									x	
3			x						x		x	
4		x		x							x	
5					x						x	
6		x	x						x		x	
7							x				x	
8		x		x				x			x	
9			x						x		x	
10		x			x					x	x	
11											x	
12		x	x	x					x		x	
13											x	
14		x					x				x	
15			x		x				x		x	
16		x		x				x			x	
17											x	
18		x	x						x		x	
19											x	
20		x		x	x					x	x	
21			x				x		x		x	
22		x									x	
23											x	
24		x	x	x				x	x		x	
25					x						x	
26		x									x	
27			x						x		x	
28		x		x			x				x	
29											x	
30		x	x		x				x	x	x	
31											x	
32		x		x				x			x	
33			x						x		x	
34		x									x	
35					x		x				x	
36		x	x	x					x		x	
37											x	
38		x									x	
39			x						x		x	
40		x		x	x			x		x	x	
41											x	
42		x	x				x		x		x	
43											x	
44		x		x							x	

Frame	Channels											
	1	2	3	4	5	6	7	8	9	10	11	12
45			x		x				x		x	
46		x									x	
47											x	
48		x	x	x				x	x		x	
49							x				x	
50		x			x					x	x	
51			x						x		x	
52		x		x							x	
53											x	
54		x	x						x		x	
55					x						x	
56		x		x			x	x			x	
57			x						x		x	
58		x									x	
59											x	
60		x	x	x	x				x	x	x	
61											x	
62		x									x	
63			x				x		x		x	x
64												
65												
66	x	x	x	x	x			x	x	x	x	
67											x	
68		x									x	
69			x						x		x	
70		x		x							x	
71					x						x	
72		x	x						x		x	
73							x				x	
74		x		x				x			x	
75			x						x		x	
76		x			x					x	x	
77											x	
78		x	x	x					x		x	
79											x	
80		x					x				x	
81			x		x				x		x	
82		x		x				x			x	
83											x	
84		x	x						x		x	
85											x	
86		x		x	x					x	x	
87			x				x		x		x	
88		x									x	
89											x	

Frame	Channels											
	1	2	3	4	5	6	7	8	9	10	11	12
90		x	x	x				x	x		x	
91					x						x	
92		x									x	
93			x						x		x	
94		x		x				x			x	
95											x	
96		x	x		x				x	x	x	
97											x	
98		x		x				x			x	
99			x						x		x	
100		x									x	
101					x			x			x	
102		x	x	x					x		x	
103											x	
104		x									x	
105			x						x		x	
106		x		x	x			x		x	x	
107											x	
108		x	x					x		x	x	
109											x	
110		x		x							x	
111			x		x					x	x	
112		x									x	
113											x	
114		x	x	x				x	x		x	
115								x			x	
116		x			x					x	x	
117			x						x		x	
118		x		x							x	
119											x	
120		x	x						x		x	
121					x						x	
122		x		x				x	x		x	
123			x						x		x	
124		x									x	
125											x	
126		x	x	x	x					x	x	
127											x	
128		x									x	
129			x					x		x	x	x
130												
131												
132	x	x	x	x	x	x	x	x	x	x	x	
133											x	
134		x									x	

Figure 1-I-1. Carriage Control Tape Format

CARRIAGE CONTROL ARRAYS FOR 580 PFC PRINTERS

Line spacing on a 580 programmable format control (PFC) line printer is controlled by a PFC array that acts as a software version of the carriage control tape in non-PFC printers. The printer has four released PFC arrays, two for six-line-per-inch print density and two for eight-line-per-inch print density. The user can specify one of the released arrays with the SC parameter on the ROUTE statement. The released arrays are listed in the NOS System Maintenance Reference Manual. If validated (refer to LIMITS Statement, section 6), the user can include a PFC array in the print file to control its spacing until another PFC array or the end of the print file is reached. This user-supplied PFC array begins with the carriage control character V.

NOTE

The PFC array does not change the print density. Print density is selected by an S or T carriage control character.

Upon reading a V carriage control character, the line printer page ejects. If the printer is not a PFC printer, the rest of the line is ignored. If the user is not validated to use the V carriage control character, the print file terminates, and the user is informed by a message in his output file.

NOTE

The V carriage control character is ineffective when a print file is routed from a time-sharing job.

PFC ARRAY SYNTAX

Carriage control character V is in column 1 of the line. The character in column 2 determines which PFC array is changed. Its legal values are the following:

Value	Description
6	Six-line-per-inch spacing. The entire array specification is on this line.
8	Eight-line-per-inch spacing. The entire array specification is on this line.
C	Eight-line-per-inch spacing. The array specification is continued on a second line. The second line begins in column 3. Columns 1 and 2 are ignored.

The PFC array specifications start in column 3. They are alphabetic characters A through L, O, and blank and have the following significance.

Character	Significance
A	Specifies top of forms code. This must be the first character in the array.
B through K	Specifies format channels 2 through 11,† respectively.
L	Specifies bottom of forms code.
O	Signals the end of the array specification. This must be the last character in the array. It has no effect on printing.
Blank	No channel specified.

NOTE

The user must specify each channel referenced in the print file in the PFC array. A channel specification can be repeated.

A six-line-per-inch array specification may be a maximum of 132 characters plus the array terminator. An eight-line-per-inch array specification may be 176 characters plus the array terminator.

If the array contains an illegal character, the array line is printed and the print file is terminated. Other invalid arrays are ignored and the file is printed using the carriage control array previously loaded into the printer.

Examples:

The following arrays are invalid.

Invalid Array	Reason
V6BCD O	Does not begin with an A.
VBA C DEO	Second character is not 6, 8, or C.
V8ABWCO	Contains an illegal character (W).

The following example uses a PFC array to print a short special form.

A programmer wants to print on the top, fifth, and bottom lines of an eight-line form (eight-line-per-inch print density). He selects the print density with the T carriage control character and then loads the following PFC array.

```
Columns:  1  2  3  4  5  6  7  8  9  10  11
          V  8  A              B        L   O
```

†The A character refers to channel 1, and L refers to channel 12. In the maximum length array specification (132 for six lines per inch and 176 for eight lines per inch), the J character can be specified only in the last position (immediately before the O).

The eight-line-per-inch carriage control array is changed as follows (x denotes a bit set):

	Channels											
	1	2	3	4	5	6	7	8	9	10	11	12
1	x											
2												
3												
4												
5		x										
6												
7												
8												x

Because only 3 of the 12 format channels are specified in the array, the other 9 channels (in the eight-line-per-inch array) remain unchanged from their last setting.

After loading the array, output lines beginning with an 8 (format channel 1) are printed at the top of form, lines beginning with a 7 (format channel 2) are printed at the fifth line, and lines beginning with a 2 (format channel 12) are printed at the bottom of the form.

OBSOLETE TAPE FORMATS J

The B, E, and X format coded tapes are no longer supported by NOS. The following description of the physical and logical characteristics of the obsolete tapes is included as a reference for TCOPY control statement users.

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	Any block containing fewer than 18 frames is considered noise, and is, therefore, ignored.
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.

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 B-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 6 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.

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.
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 eight frames for seven-track tapes or six frames for nine-track tapes is considered noise, and is, therefore, ignored.
Special considerations	• X-formatted tapes cannot be labeled. • All nine-track tapes are written in an even multiple of bytes.

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.

Option	PO= Option	Description
1	I	Rewrites the block on which the end-of-tape condition is sensed as the first block on the next volume, if the system senses the EOT during a write operation. During a read operation, the block on which the EOT occurred is ignored and reading continues on the next volume. If a tape mark and the EOT are sensed at the same time, the EOT is ignored.

Option	PO= Option	Description
2	P	Writes a trailer sequence following the block on which the EOT is sensed, if the system senses the EOT during a write operation. Any data that occurs following the block on which EOT is sensed, yet before the tape mark, is ignored. During a read operation, the system transfers the block on which the EOT is sensed to the user job. The read operation resumes on the next reel. If a tape mark and the EOT are sensed at the same time, the EOT is ignored.

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.

INDEX

COMMENT SHEET

MANUAL TITLE: CDC NOS Version 1 Reference Manual, Volume 1

PUBLICATION NO.: 60435400 **REVISION:** M

NAME: _____

COMPANY: _____

STREET ADDRESS: _____

CITY: _____ **STATE:** _____ **ZIP CODE:** _____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

CUT ALONG LINE

AA3419 REV. 4/79 PRINTED IN U.S.A.

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

FOLD ON DOTTED LINES AND STAPLE

TAPLE

STAPLE

OLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55112

CUT ALONG LINE

OLD

FOLD

)RPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINN. 55440
\LES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.A.

CONTROL DATA CORPORATION