

CDC® VSOS VERSION 2

FOR USE WITH
CYBER 200 SERIES
COMPUTER SYSTEM

Volume 1 of 2

REFERENCE MANUAL



VSOS CONTROL STATEMENT INDEX

This index lists each VSOS control statement and interactive request line and the number of the page on which it is described.

ATTACH	4-13	PCREATE	4-100
AUDIT	4-15	PDELETE	4-101
BEGIN	4-6.3	PDESTROY	4-102
CHARGE	4-21	PDETACH	4-102
COMMENT	4-22	PERMIT	4-103
COMPARE	4-23	PFILES	4-105
COPY	4-25	PROC	4-6.3
COPYL	4-28	PROCEED	4-106
DAYFILE	4-30	PURGE	4-106.1
DEFINE	4-30.1	Q	4-107
DEBUG	6-2	REQUEST	4-112.1
DIVERT	4-34	RERUN	4-120
DMAP	4-34.2	RESOURCE	4-121
DROP	4-36	RETURN	4-124
DUMP	6-23	REWIND	4-126
DUMPF	4-36.2	SET	4-127
EDITPUB	4-46	SKIP	4-128.1
ELSE	4-6.2	SLGEN	4-129
ENDIF	4-6.2	SUBMIT	4-132
EXIT	4-48	SUMMARY	4-134
FILES	4-49	SWITCH	4-136
GIVE	4-53	TASKATT	4-139
IF	4-6.1	TV	4-140
LABEL	4-55	UPDATE	5-1
LIMITS	4-60.1	USER	4-142
LISTAC	4-61	\$BB	3-8
LOAD	4-65	\$BYE	3-10.1
LOADPF	4-80	\$HELLO	3-10.1
LOGIN	3-6	\$I	3-10
LOOK	6-13	\$LC	3-9
MFGIVE	3-25	\$OP	3-10
MFLINK	4-88	\$P	3-8
MFQUEUE	4-91	\$PR	3-8
MFTAKE	3-25	\$S	3-8
NORERUN	4-93	\$SU	3-8
OLE	4-94	\$T	3-8
PACCESS	4-97	\$UC	3-9
PASSWORD	4-98	\$X	3-9
PATTACH	4-99	\$?	3-8

CDC® VSOS VERSION 2

**FOR USE WITH
CYBER 200 SERIES
COMPUTER SYSTEM**

Volume 1 of 2

REFERENCE MANUAL



REVISION RECORD

REVISION	DESCRIPTION
A (04-16-82)	Manual released.
B (10-15-82)	Manual revised to reflect VSOS 2.0 CCR changes (level 575). New features documented include the GDWC LOAD parameter, individual access permission sets, and conversion routines for CYBER 170 arithmetic data formats.
C (07-29-83)	Manual revised to reflect VSOS 2.1 PSR level 592 changes. New features include on-line magnetic tape support, the DAYFILE control statement, and the Q5MEMORY subroutine. Because extensive changes have been made, change bars and dots are not used and all pages reflect the latest revision level. This edition obsoletes all previous revisions.
D (03-30-84)	Manual revised to reflect VSOS 2.1.5 PSR level 607 changes. New features include the SUBMIT control statement, additions to the DEBUG directives and additions to the error messages. Due to extensive changes, change bars and dots are not used and all pages reflect the latest revision level. This edition obsoletes all previous editions.
E (10-31-85)	Manual revised to reflect VSOS 2.2 PSR level 644 changes. New features include project tracking, small job throughput, multiple batch jobs per user, dynamic file allocation and device overflow, system channel expander, and rejected queue files. CYBER 200 FORTRAN is no longer supported. Due to extensive changes, change bars and dots are not used and all pages reflect the latest revision level. This edition obsoletes all previous editions.
F (04-18-86)	Manual revised to reflect VSOS 2.2.5 PSR level 654 changes. Changes documented are small job throughput improvement, explicit I/O performance, automatic job category selection and job pre-abort, MFQUEUE improvements, on-line DUMPF, purge files by access date, and LIMITS control statements for restricting tape usage via validation.
G (12-05-86)	Manual revised to reflect VSOS 2.3 PSR level 670. New features described include drop file map overflow reduction and on/off RHF NADs. This revision also includes updates to control statements, SIL calls, and messages. This edition obsoletes all previous editions.
H (10-23-87)	Manual revised to reflect VSOS 2.3.5 at PSR level 690. This revision documents managing production files at security-sensitive sites, and the new ELSE, ENDIF, IF, DIVERT, and DROP control statements. It also includes updates to existing control statements and SIL subroutines/routines, and changes to the queue file transfer procedures.
J (11-15-88)	Manual revised to reflect VSOS 2.3.7 at PSR level 712.
Publication No. 60459410	

REVISION LETTERS I, O, Q, S, X AND Z ARE NOT USED.

Address comments concerning this manual to:

Control Data Corporation
Technology and Publications Division
4201 North Lexington Avenue
St. Paul, Minnesota 55126-6198

© 1982, 1983, 1984, 1985, 1986, 1987, 1988
by Control Data Corporation
All rights reserved
Printed in the United States of America

or use Comment Sheet in the back of this manual.

LIST OF EFFECTIVE PAGES

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

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
Front Cover	-	2-31	E	4-11	H	4-57	H	4-104	E
Inside Front Cover	J	2-32	E	4-12	E	4-58	F	4-105	G
Title Page	-	2-33	E	4-13	E	4-59	F	4-106	G
2	J	2-34	F	4-14	E	4-60	F	4-106.1/	
3	J	2-35	J	4-15	H	4-60.1/4-60.2	H	4-106.2	G
4	J	2-36	J	4-16	H	4-61	E	4-107	E
5/6	J	2-36.1/2-36.2	J	4-16.1	H	4-62	E	4-108	H
7	F	2-37	G	4-16.2	H	4-63	H	4-108.1/	
8	E	2-38	E	4-17	H	4-64	E	4-108.2	H
9/10	E	3-1	H	4-18	H	4-65	H	4-109	G
11	J	3-2	E	4-19	H	4-66	G	4-110	H
12	J	3-3	E	4-20	H	4-67	E	4-111	J
13	J	3-4	H	4-21	E	4-68	E	4-112	J
14	J	3-5	E	4-22	G	4-69	F	4-112.1/	
15	J	3-6	H	4-23	J	4-70	F	4-112.2	J
16	J	3-7	G	4-24	F	4-71	G	4-113	J
17	J	3-8	G	4-25	J	4-72	H	4-114	J
18	J	3-9	J	4-26	J	4-73	H	4-115	J
19	J	3-10	G	4-27	G	4-74	G	4-116	J
20	J	3-10.1/3-10.2	G	4-28	J	4-74.1	J	4-116.1	J
1-1	E	3-11	J	4-29	J	4-74.2	J	4-116.2	J
1-2	E	3-12	F	4-30	G	4-74.3	J	4-117	F
1-3	G	3-13	J	4-30.1/4-30.2	G	4-75	J	4-118	E
1-4	G	3-14	J	4-31	F	4-76	H	4-119	J
1-5	G	3-15	J	4-32	G	4-77	H	4-120	E
1-6	F	3-16	G	4-33	G	4-78	H	4-121	J
2-1	E	3-17	G	4-34	H	4-78.1/4-78.2	J	4-122	J
2-2	E	3-18	G	4-34.1	H	4-79	J	4-123	E
2-3	H	3-19	F	4-34.2	J	4-80	J	4-124	E
2-4	E	3-20	F	4-35	H	4-81	J	4-125	E
2-5	E	3-21	F	4-36	H	4-82	J	4-126	E
2-6	E	3-22	J	4-36.1	H	4-82.1	J	4-127	H
2-7	J	3-22.1/3-22.2	H	4-36.2	H	4-82.2	J	4-128	H
2-8	E	3-23	F	4-36.3	H	4-83	J	4-128.1/	
2-9	H	3-24	G	4-37	J	4-84	H	4-128.2	G
2-10	E	3-25	F	4-38	H	4-84.1/4-84.2	J	4-129	F
2-11	E	3-26	H	4-38.1	J	4-85	G	4-130	F
2-12	E	3-27	E	4-38.2	J	4-86	H	4-131	E
2-13	E	3-28	G	4-39	J	4-86.1/4-86.2	H	4-132	H
2-14	E	3-29	H	4-40	H	4-87	G	4-133	H
2-15	H	3-30	G	4-40.1/4-40.2	H	4-88	G	4-134	E
2-16	F	3-31	J	4-41	E	4-88.1/4-88.2	J	4-135	E
2-17	J	3-32	G	4-42	J	4-89	J	4-136	E
2-18	E	3-33	G	4-43	G	4-90	E	4-137	E
2-19	E	4-1	H	4-44	H	4-91	J	4-138	E
2-20	E	4-2	H	4-44.1/4-44.2	H	4-92	J	4-139	G
2-21	E	4-2.1/4-2.2	H	4-45	G	4-92.1/4-92.2	J	4-140	E
2-22	E	4-3	H	4-46	E	4-93	E	4-141	E
2-23	E	4-4	F	4-47	E	4-94	F	4-142	J
2-24	G	4-5	F	4-48	G	4-95	F	5-1	E
2-25	E	4-6	H	4-49	E	4-96	F	5-2	E
2-26	F	4-6.1	H	4-50	G	4-97	E	5-3	E
2-27	G	4-6.2	H	4-51	H	4-98	E	5-4	G
2-28	G	4-6.3/4-6.4	J	4-52	E	4-99	E	5-5	J
2-28.1/2-28.2	G	4-7	E	4-53	E	4-100	E	5-6	F
2-29	E	4-8	E	4-54	G	4-101	E	5-7	G
2-30	G	4-9	E	4-55	E	4-102	F	5-8	G
		4-10	E	4-56	E	4-103	H	5-9	G

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
5-10	E	8-13	G	8-84	E	9-58	J	9-58	J
5-11	E	8-14	J	8-85	E	9-59	E	9-59	E
5-12	E	8-15	H	8-86	E	9-60	J	9-60	J
5-13	E	8-16	J	8-87	E	9-61	J	9-61	J
5-14	E	8-17	J	8-88	E	9-62	J	9-62	J
5-15	E	8-18	G	8-89	H	9-63	E	9-63	E
5-16	E	8-19	F	8-90	E	9-64	E	9-64	E
5-17	E	8-20	F	8-91	E	9-65	J	9-65	J
5-18	E	8-21	J	8-92	E	9-66	J	9-66	J
5-19	E	8-22	J	8-93	E	9-67	E	9-67	E
5-20	J	8-23	H	8-94	E	9-68	E	9-68	E
5-21	J	8-24	H	8-95	E	9-69	E	9-69	E
5-21.1	J	8-25	H	8-96	E	9-70	E	9-70	E
5-21.2	J	8-26	H	9-1	E	9-71	J	9-71	J
5-22	J	8-27	H	9-2	E	9-72	E	9-72	E
5-22.1/5-22.2	J	8-28	H	9-3	E	9-73	E	9-73	E
5-23	E	8-29	E	9-4	H	9-74	E	9-74	E
5-24	F	8-30	H	9-5	E	9-75	J	9-75	J
5-25	F	8-31	E	9-6	H	9-76	J	9-76	J
5-26	F	8-32	E	9-7	E	9-77	J	9-77	J
5-26.1/5-26.2	G	8-33	E	9-8	G	9-78	J	9-78	J
5-27	G	8-34	E	9-9	E	9-79	J	9-79	J
5-28	E	8-35	E	9-10	E	9-80	F	9-80	F
5-29	E	8-36	E	9-11	J	9-81	G	9-81	G
5-30	E	8-37	E	9-12	H	9-82	G	9-82	G
6-1	E	8-38	E	9-13	J	9-82.1/9-82.2	J	9-82.1/9-82.2	J
6-2	H	8-39	G	9-14	J	9-83	E	9-83	E
6-2.1/6-2.2	H	8-40	G	9-15	H	9-84	G	9-84	G
6-3	J	8-41	E	9-16	H	9-85	E	9-85	E
6-4	J	8-42	E	9-17	E	9-86	G	9-86	G
6-5	J	8-43	E	9-18	E	9-87	E	9-87	E
6-6	J	8-44	G	9-19	E	9-88	E	9-88	E
6-7	J	8-45	G	9-20	E	9-89	H	9-89	H
6-8	J	8-46	E	9-21	E	9-90	J	9-90	J
6-9	J	8-47	E	9-22	G	9-91	E	9-91	E
6-10	J	8-48	G	9-22.1/9-22.2	G	9-92	E	9-92	E
6-11	J	8-49	J	9-23	G	9-93	E	9-93	E
6-12	J	8-50	E	9-24	E	9-94	E	9-94	E
6-12.1	J	8-51	J	9-25	E	9-95	E	9-95	E
6-12.2	J	8-52	H	9-26	H	9-96	E	9-96	E
6-12.3/6-12.4	J	8-53	E	9-27	G	9-97	E	9-97	E
6-13	E	8-54	H	9-28	H	9-98	E	9-98	E
6-14	E	8-55	H	9-29	G	9-99	E	9-99	E
6-15	E	8-56	E	9-30	E	9-100	G	9-100	G
6-16	E	8-57	H	9-31	E	9-101	E	9-101	E
6-17	H	8-58	J	9-32	F	9-102	F	9-102	F
6-18	J	8-59	E	9-33	F	9-103	H	9-103	H
6-19	E	8-60	J	9-34	F	9-104	E	9-104	E
6-20	E	8-61	E	9-35	F	9-105	G	9-105	G
6-21	E	8-62	E	9-36	E	9-106	E	9-106	E
6-22	E	8-63	J	9-37	J	9-107	E	9-107	E
6-23	G	8-64	E	9-38	J	9-108	G	9-108	G
6-24	E	8-65	J	9-39	H	9-109	E	9-109	E
7-1	E	8-66	J	9-40	G	9-110	G	9-110	G
7-2	E	8-67	E	9-41	J	9-110.1/ 9-110.2	G	9-110.1/ 9-110.2	G
7-3	E	8-68	E	9-42	J	9-111	E	9-111	E
7-4	E	8-69	H	9-43	G	9-112	F	9-112	F
7-5	E	8-70	E	9-44	E	9-113	F	9-113	F
7-6	E	8-71	E	9-45	E	9-114	F	9-114	F
8-1	E	8-72	E	9-46	G	9-115	J	9-115	J
8-2	E	8-73	F	9-47	E	9-116	J	9-116	J
8-3	E	8-74	J	9-48	H	9-117	H	9-117	H
8-4	E	8-75	E	9-49	E	9-118	E	9-118	E
8-5	E	8-76	E	9-50	E	9-119	E	9-119	E
8-6	F	8-77	E	9-51	G	9-120	J	9-120	J
8-7	E	8-78	E	9-52	G	9-121	J	9-121	J
8-8	E	8-79	E	9-53	G	9-122	E	9-122	E
8-9	E	8-80	E	9-54	G	9-123	J	9-123	J
8-10	E	8-81	E	9-55	E	9-124	E	9-124	E
8-11	H	8-82	E	9-56	G	9-125	E	9-125	E
8-12	H	8-83	E	9-57	J				

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
B-14	H	B-85	J	F-15	G				
B-15	H	B-86	J	G-1	J				
B-16	H	B-87	J	G-2	J				
B-17	H	B-88	J	G-3	J				
B-18	H	B-89	J	G-4	J				
B-19	H	B-90	J	Index-1	J				
B-20	H	B-91	J	Index-2	J				
B-21	H	B-92	J	Index-3	J				
B-22	H	B-93	J	Index-4	J				
B-23	H	B-94	J	Index-5	J				
B-24	H	B-95	J	Index-6	J				
B-25	H	B-96	J	Index-7	J				
B-26	J	B-97	J	Index-8	J				
B-27	J	B-98	J	Index-9	J				
B-28	J	B-99	J	Index-10	J				
B-29	J	B-100	J	Index-11	J				
B-30	J	B-101	J	Index-12	J				
B-31	J	B-102	J	Comment Sheet	J				
B-32	J	B-103	J	Inside Back					
B-33	J	B-104	J	Cover	J				
B-34	J	B-105	J	Back Cover	-				
B-35	J	B-106	H						
B-36	J	B-107	H						
B-37	J	B-108	H						
B-38	J	B-109	H						
B-39	J	B-110	J						
B-40	J	B-111	J						
B-41	J	B-112	J						
B-42	J	C-1	E						
B-43	J	C-2	H						
B-44	J	C-3	H						
B-45	J	C-4	H						
B-46	J	C-5	H						
B-47	J	C-6	H						
B-48	J	C-7	H						
B-49	J	C-8	H						
B-50	J	C-9	J						
B-51	J	C-10	J						
B-52	J	C-11	J						
B-53	J	D-1	E						
B-54	J	D-2	E						
B-55	J	D-3	E						
B-56	J	D-4	F						
B-57	J	D-5	J						
B-58	J	D-6	E						
B-59	J	D-7	G						
B-60	J	D-8	E						
B-61	J	D-9	E						
B-62	J	E-1	E						
B-63	J	E-2	E						
B-64	J	E-3	E						
B-65	J	E-4	E						
B-66	J	E-5	E						
B-67	J	E-6	E						
B-68	J	E-7	E						
B-69	J	E-8	E						
B-70	J	E-9	E						
B-71	J	F-1	E						
B-72	J	F-2	E						
B-73	J	F-3	E						
B-74	J	F-4	J						
B-75	J	F-5	E						
B-76	J	F-6	E						
B-77	J	F-7	J						
B-78	J	F-8	F						
B-79	J	F-9	F						
B-80	J	F-10	J						
B-81	J	F-11	E						
B-82	J	F-12	E						
B-83	J	F-13	J						
B-84	J	F-14	J						

PREFACE

This manual describes the CONTROL DATA® Virtual Storage Operating System (VSOS) for the CDC® CYBER 200 Series Computer System. This manual is published in two volumes:

- Volume 1 describes system utilities and system interface language (SIL) subroutines. It also contains a general description of CYBER 200 hardware and operating system software, file concepts, and task execution. It is written primarily for the applications programmer.
- Volume 2 describes system messages and job management tables. It also describes system accounting file formats, common execute line routines, and loader conventions. It is written primarily for the system programmer.

RELATED PUBLICATIONS

Related information can be found in the following publications:

<u>Control Data Publication</u>	<u>Publication Number</u>
VSOS Version 2, Reference Manual, Volume 2	60459420
FORTTRAN 200 Version 1 Reference Manual	60480200
CYBER 200 Maintenance Software System Reference Manual	60457200
CYBER 200 Assembler Version 2 Reference Manual	60485010
CYBER 200/Model 205 Computer System Hardware Reference Manual	60456020
CYBER 200/Model 205 Troubleshooting Guide	60430060
VSOS Version 2 Operator's Guide	60459430
VSOS Version 2 Installation Handbook	60459440
Remote Host Facility Handbook (Use with NOS system)	60459060
Remote Host Facility Handbook for IBM System (Use with MVS/JES2, MVS/JES3, and MVS/ASP systems)	60459050
Remote Host Facility Handbook (Use with SCOPE 2 system)	60455610
Remote Host Facility User's Guide	60460620
VSOS Version 2 Site Manager's Handbook	60461490
VSOS User's Guide for FORTTRAN 200 Programmers	60455390

Control Data manuals can be ordered from:

Literature and Distribution Services
STP005
304 North Dale Street
St. Paul, MN 55103

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 parameters.

Control Data no longer supports the FORTRAN 66 compiler.

NOTATIONS USED IN THIS MANUAL

UPPERCASE	Words or character strings that must be entered as shown. They must be spelled correctly, including any = or / shown.
<u>UNDERLINED UPPERCASE</u>	Words or character strings that can be abbreviated to the number of underlined characters.
Lowercase words	Generic terms that represent the parameters or character strings supplied by the programmer. When generic terms are repeated in a format, a number or letter might be appended.
[] Brackets	An optional portion of a format. All parameters enclosed within the brackets can be omitted at the programmer's option. The brackets are editorial conventions only; they are not part of the format.
{ } Braces	A portion of a format in which only one of the vertically stacked items can be used. The braces are editorial conventions only; they are not part of the format.
. . . Ellipses	An indicator of repetition. The portion of the format immediately preceding the ellipses can be repeated at the programmer's option.
△	An indicator of a blank. In a format, this character indicates that a blank or space should appear.
#	An indicator that hexadecimal numbers follow. Numbers used in this manual are decimal unless noted as hexadecimal.

Punctuation characters shown within formats are required unless the text indicates that another punctuation character can be substituted.

CONTENTS

<p>1. INTRODUCTION 1-1</p> <p>System Configuration 1-1</p> <p style="padding-left: 20px;">CYBER 200 Mainframe 1-2</p> <p style="padding-left: 20px;">CYBER 200 Memory 1-2</p> <p style="padding-left: 20px;">Maintenance and Control Unit 1-2</p> <p style="padding-left: 20px;">Loosely Coupled Network (LCN) 1-3</p> <p>Operating System 1-4</p> <p style="padding-left: 20px;">Resident System 1-4</p> <p style="padding-left: 20px;">Virtual System 1-4</p> <p style="padding-left: 20px;">Privileged System Tasks 1-4</p> <p>VSOS User Interface 1-5</p> <p>Remote Host Facility (RHF) 1-5</p> <p>CYBER 200 Comparison 1-5</p> <p>Virtual Memory Addressing 1-6</p> <p>Register File 1-6</p> <p>2. FILE CONCEPTS 2-1</p> <p>File Attributes 2-1</p> <p>File Types 2-2</p> <p style="padding-left: 20px;">Controllee Files 2-2</p> <p style="padding-left: 20px;">Data Files 2-2</p> <p style="padding-left: 20px;">Drop Files 2-2</p> <p style="padding-left: 40px;">Drop File Naming Convention 2-3</p> <p style="padding-left: 40px;">Restarting a Task 2-3</p> <p style="padding-left: 40px;">Bound Explicit and Implicit Maps 2-4</p> <p style="padding-left: 20px;">Output Files 2-4</p> <p style="padding-left: 40px;">Print Files 2-5</p> <p style="padding-left: 40px;">Print Control Characters 2-5</p> <p>Output File Error Processing 2-7</p> <p>File Duration 2-8</p> <p style="padding-left: 20px;">Scratch Files 2-8</p> <p style="padding-left: 20px;">Local Files 2-8</p> <p style="padding-left: 20px;">Permanent Files 2-8</p> <p>File Usage Controls 2-9</p> <p style="padding-left: 20px;">File Security Levels 2-9</p> <p style="padding-left: 20px;">File Patterning 2-9</p> <p style="padding-left: 20px;">File Ownership 2-9</p> <p style="padding-left: 40px;">File Search Hierarchy 2-12</p> <p style="padding-left: 40px;">Private Files 2-12</p> <p style="padding-left: 40px;">Pool Files 2-12</p> <p style="padding-left: 40px;">System Pool 2-13</p> <p style="padding-left: 40px;">Public Files 2-13</p> <p>File Access Permissions 2-14</p> <p style="padding-left: 20px;">Read Permission 2-14</p> <p style="padding-left: 20px;">Write Permission 2-14</p> <p style="padding-left: 20px;">Append Permission 2-14</p> <p style="padding-left: 20px;">Modify Permission 2-15</p>	<p>Execute Permission 2-15</p> <p>Access Permission Sets 2-15</p> <p>Concurrent File Access 2-16</p> <p>File I/O 2-17</p> <p style="padding-left: 20px;">Explicit I/O 2-17</p> <p style="padding-left: 20px;">Implicit I/O 2-17</p> <p>Logical File Structures 2-18</p> <p style="padding-left: 20px;">Logical Record Formats 2-18</p> <p style="padding-left: 40px;">ANSI Fixed Length (F) Record Format 2-18</p> <p style="padding-left: 40px;">Record Mark Delimited (R) Record Format 2-19</p> <p style="padding-left: 40px;">Undefined Structure (U) Record Format 2-19</p> <p style="padding-left: 40px;">Control Word Delimited (W) Record Format 2-20</p> <p style="padding-left: 20px;">CYBER Record Manager Control</p> <p style="padding-left: 40px;">Word (L) Record Format 2-21</p> <p style="padding-left: 40px;">System Block (B) Record Format 2-22</p> <p>Blocking Types 2-23</p> <p style="padding-left: 20px;">C Blocking 2-23</p> <p style="padding-left: 20px;">I Blocking 2-23</p> <p style="padding-left: 20px;">K Blocking 2-24</p> <p>File Organization 2-25</p> <p style="padding-left: 20px;">Sequential Access Organization 2-25</p> <p style="padding-left: 20px;">Direct Access Organization 2-25</p> <p>Device Characteristics 2-26</p> <p>Mass Storage Files 2-26</p> <p style="padding-left: 20px;">File Space Allocation 2-26</p> <p>Tape Files 2-28.1</p> <p style="padding-left: 20px;">Tape Drive Reservation 2-28.1</p> <p style="padding-left: 20px;">Volume Assignment 2-29</p> <p style="padding-left: 20px;">Volume Switching 2-29</p> <p style="padding-left: 20px;">Tape Labeling 2-29</p> <p style="padding-left: 20px;">Tape Data Recording 2-32</p> <p style="padding-left: 20px;">Tape Data Organization 2-33</p> <p style="padding-left: 20px;">Tape Error Processing 2-36.1</p> <p style="padding-left: 20px;">User Error Processing 2-37</p> <p>Connected Interactive Terminal Files 2-37</p> <p>3. TASK EXECUTION 3-1</p> <p>Initiating Controllee Execution 3-1</p> <p>Virtual Space Mapping 3-2</p> <p>Controllee Chains 3-3</p> <p>System Access 3-4</p> <p style="padding-left: 20px;">User Validation 3-4</p> <p style="padding-left: 20px;">Interactive System Access 3-5</p> <p style="padding-left: 20px;">VSOS Interactive Login 3-5</p>
--	---

Batch System Access	3-6	Conditional Statement Processing	4-6.2
Interactive Session	3-7	Control Statement Procedures	4-6.3
Break Character	3-7	PROC Statement	4-6.3
Interactive Request Lines	3-7	BEGIN Statement	4-6.3
Changing the Interactive		Control Statement Execution	
Request Special Character	3-7	Sequence	4-7
Terminal Information Requests	3-8	Procedure Nesting	4-8
Case Conversion Request	3-9	Formal Parameter Substitution	4-8
Operator Message Request	3-10	Matching Substitution Values	
Task Interrupt Request	3-10	to Formal Parameters	4-9
Session Termination Request	3-10.1	Omitting Substitution Values	4-10
Interactive Execute Line	3-10.1	Suppressing Formal Parameter	
Task Data Input	3-12	Substitution	4-11
Dynamic and Static Execution	3-12	Concatenating Substitution	
Batch Job	3-14	Values	4-11
Batch Input File Structure	3-14	Suppressing @ or ^ Character	
Batch Control Statement	3-14	Removal	4-12
Job Scheduling	3-15	ATTACH - Attach Permanent Files	4-13
Job Processing	3-16	AUDIT - List File Information	4-15
Job Dayfile	3-17	File Specification	4-17
Job Termination	3-17	AUDIT Output	4-18
Job Termination Procedure	3-18	CHARGE - Assign Account and Project	
Job Abort	3-18	Number	4-21
Abnormal Job Termination	3-18	COMMENT - Send Message to Job Dayfile	4-22
Job Processing Example	3-18	COMPARE - Compare File Contents	4-23
Remote Host Facility	3-21	Controllee File Comparison	4-24
Interactive Access	3-21	COPY - Copy a File	4-25
Queue File Transfers	3-22	Copying to or from a Tape File	4-26
CYBER 200 Job Submission	3-22	Copying to a Mass Storage File	4-26
Output File Routing	3-23	Controllee File Copy	4-27
Explicit File Routing	3-23	COPYL - Copy Logical Records	4-28
RHF Permanent File Requests	3-24	DAYFILE - Copy the Job Dayfile	4-30
Permanent File Requests	3-24	DEFINE - Define a Permanent File	4-30.1
Permanent File Audit Request	3-25	Defining a New File	4-33
Direct Access File Transfers	3-25	DIVERT - Change the Destination of	
File Archiving	3-27	an Output File	4-34
Task Termination	3-28	DMAP - Provide Information on	
User Reprieve	3-28	Location of File Segments	4-34.2
Abnormal Termination Control	3-28	DROP - Remove a Job from a Queue	4-36
ATC Interrupt Subroutine	3-29	DUMPF - Archive Files	4-36.2
Enabling and Disabling ATC	3-30	Specification of Files to be	
Resource Allocation	3-31	Archived	4-40.1
Batch Resource Limits	3-31	Archive File Format	4-41
Interactive Resource Limits	3-32	Archiving to a Front-End System	4-42
Accounting	3-33	Archiving to CYBER 200 Mass	
		Storage	4-42
		Using DUMPF to Purge Files	4-43
		Archiving to CYBER 200 On-Line	
		Tapes	4-43
		DUMPF Output	4-43
4. CONTROL STATEMENTS	4-1	EDITPUB - Add or Destroy Public	
Control Statement Parameter Format	4-4	File	4-46
Interactive Control Statement		Variable Rate Index Specification	4-47
Execution	4-5	EXIT - Set Abnormal Termination Path	4-48
Control Statement Management	4-6	FILES - List File Information	4-49
Control Statement Variables	4-6	FILES Output	4-51
Conditional Control Statements	4-6.1	Interactive Utility Execution	4-52
IF Control Statement	4-6.1	GIVE - Change File Owner	4-53
ELSE Control Statement	4-6.2	LABEL - Label Tape File	4-55
ENDIF Control Statement	4-6.2	Multifile Sets	4-58

Writing a Multifile Set	4-58	Input Queue Status	4-108.1
Reading a Multifile Set	4-59	Executing Task Status	4-109
Rewriting Files in a Multifile Set	4-59	Output File Status	4-110
LIMITS - List User Validations	4-60.1	REQUEST - Create Local File	4-112.1
LISTAC - List Access Permission Sets	4-61	File Space Allocation	4-117
LISTAC Output	4-63	Tape File Request	4-118
LOAD - Generate Controllee File	4-65	Tape Labels	4-118
Files Used to Generate a Controllee	4-65	Data Format Specification	4-119
Object Code Files	4-65	Processing Options	4-119
Listing File	4-66	Operator Message	4-119
Controllee File	4-66	RERUN - Set Rerun Status	4-120
Satisfying External References	4-69	RESOURCE - Set Job Resource Limits	4-121
Dynamic Linking Using the System		Tape Drive Reservation	4-123
Shared Library	4-70	RETURN - Evict Local Files or Detach Permanent Files	4-124
Dynamic Linker	4-70	Returning Tape Files	4-125
Dynamic Execution	4-71	REWIND - Rewind a Tape File	4-126
Dynamically Linked Controllees	4-71	SET - Change Job Characteristics	4-127
Page Grouping	4-71	SKIP - Reposition a Tape File	4-128.1
Grouping Controllee File Blocks	4-72	SLGEN - Construct System Shared Library	4-129
Grouping Unmapped Blocks	4-72	SUBMIT - Submit a File to a Queue	4-132
Grouping Parameter Mapping	4-73	SUMMARY - Provide Resource Usage Information	4-134
Space Initialization	4-73	SUMMARY Output	4-134
Target Page Size	4-74	SWITCH - Change File Attributes	4-136
Control Statement Format	4-74.1	TASKATT - Alter a Task's Attributes	4-139
Interactive Load Execution	4-78.1	TV - Set Threshold Value	4-140
LOADPF - Reload Files	4-80	USER - Provide User Validation Information	4-142
RHF Reloading	4-84		
Reloading from CYBER 200 Mass Storage	4-84		
Reloading from CYBER 200 On-Line Tapes	4-84.1	5. UPDATE	5-1
User Reloading Capabilities	4-85	Examples	5-2
Specification of the Files to Be Reloaded	4-85	General Processing	5-4
LOADPF Output	4-86	Update Mode and Files	5-6
MFLINK - Permanent File Transfer	4-88	Input File	5-7
Character Code Conversion	4-90	New Program Library	5-8
Logical Structure Conversion	4-90	Source File	5-8
MFQUEUE - Explicit File Routing	4-91	Old Program Library	5-8
NORERUN - Set Norerun Status	4-93	Compile File	5-8
OLE - Object Library Editor	4-94	List File	5-9
PACCESS - Authorize Pool Access	4-97	Pullmod File	5-9
PASSWORD - Change User Password	4-98	Creation of Program Library	5-9
PATTACH - Attach a Pool	4-99	Card Identification	5-10
PCREATE - Create a Pool	4-100	Correction Run	5-11
PDELETE - Remove User Access to a Pool	4-101	Deck List and Directory Order	5-12
PDESTROY - Destroy a Pool	4-102	Purge and Yank Directives	5-12
PDETACH - Detach an Attached Pool	4-102	Overlapping Corrections	5-13
PERMIT - Change Access Permission Set	4-103	Update Directives	5-14
PFILES - List Pool Information	4-105	ADDFILE Directive	5-15
Proceed - Set Abnormal Termination Path	4-106	BEFORE Directive	5-16
PURGE - Destroy Permanent or Pool Files	4-106.1	CALL Directive	5-16
Q - List Job Status	4-107	COMDECK Directive	5-17
		COMPILE Directive	5-18
		DECK Directive	5-19
		DEFINE Directive	5-20

DELETE Directive	5-20	SIL Non-I/O Calls	8-7
ENDIF Directive	5-21	Q5ADVISE - Advise System of	
IDENT Directive	5-21	Virtual Space Requirements	8-8
IF Directive	5-22	Q5CPUTIM - Get CPU Time	8-10
INSERT Directive	5-21.1	Q5DCDDST - Decode Disk Status	
MOVE Directive	5-21.1	Table	8-11
PULLMOD Directive	5-22	Q5DCDMSC - Decode Miscellaneous	
PURDECK Directive	5-22.1	Table	8-13
PURGE Directive	5-23	Q5DCDPFI - Decode Pack File Index	8-18
READ Directive	5-24	Q5DCDPLB - Decode Pack Label	8-29
WIDTH Directive	5-24	Q5DESBIF - Destroy Batch Input	
YANK Directive	5-25	File	8-31
YANKDECK Directive	5-25	Q5DISAMI - Disable Message	
/ Comment Directive	5-26	Interrupts	8-32
Update Control Statement	5-26.1	Q5DISATI - Disable Abnormal	
		Termination Control	8-33
		Q5DMPACT - Dump Cumulative	
		Accounting Buffer	8-34
6. DEBUGGING	6-1	Q5ENAMI - Enable Message	
		Interrupts	8-35
DEBUG	6-2	Q5ENATI - Enable Abnormal	
DEBUG Control Statement	6-2.1	Termination Control	8-37
DEBUG Directives	6-3	Q5GETACT - Get Resource Usage	
Dump or Display Directives	6-5	Statistics	8-38
Register Directives	6-7	Q5GETCTS - Get Controllee	
Alter Memory Directives	6-9	Termination Status	8-40
Restore Memory Directive	6-11	Q5GETIIP - Get Invisible Package	8-41
Program Control Directives	6-12	Q5GETIRF - Get Register File	8-42
LOOK	6-13	Q5GETLP - Get Large Page Limits	8-43
LOOK Control Statement	6-13	Q5GETMCE - Get Message from	
LOOK Directives	6-13	Controllee	8-44
SEARCH Directive	6-16	Q5GETMCR - Get Message from	
HSEARCH Directive	6-17	Controller	8-45
Disposition of Directive		Q5GETMOP - Get Message from	
Output	6-18	Operator	8-47
Display and Dump		Q5GETMPG - Get Minus Page	8-48
Directives	6-18	Q5GETPFI - Get Pack Label and	
Directives for Entering		File Index	8-49
Values	6-20	Q5GETTL - Get Time Limit	8-50
Declaration of Directive		Q5GETTN - Get Task Attributes	8-51
Address Type	6-21	Q5GETUID - Get User Number	8-53
DUMP	6-23	Q5INIT - Initialize Controllee	8-55
		Q5INITCH - Initialize Controllee	
		Chain	8-56
7. CHECKPOINT/RESTART	7-1	Q5LFIHIR - List File Index Entry	
Checkpointing a Task	7-1	By Hierarchical Search	8-58
Task Processing After the		Q5LFIPOI - List Pool File Indices	8-60
CHKPNT Call	7-2	Q5LFIPIRI - List Private File	
Restarting a Task	7-5	Indices	8-63
Restarting a Task That Uses		Q5LFIPIB - List Public File	
Tape Files	7-5	Indices	8-66
		Q5LSTBUT - List Bank Update Table	8-68
		Q5LSTCH - List Controllee Chain	8-69
		Q5LSTSTB - List Statistics Buffer	8-71
		Q5LSTTCB - List Timecard Buffer	8-72
		Q5MEMORY - Allocate Static Stack	8-73
		Q5RECALL - Suspend Task Execution	8-74
		Q5REPREV - Enable or Disable User	
		Reprieve	8-75
8. SYSTEM INTERFACE LANGUAGE			
(NON-I/O CALLS)	8-1		
Overview	8-2		
SIL Error Processing	8-5		
SIL Call Format	8-6		
No Operation Keywords	8-7		

Q5RFI - Return from Interrupt Subroutine	8-77	Opening a File for Implicit I/O	9-48
Q5RUNBIF - Rerun Batch Input File	8-78	Files Connected to Terminals	9-48
Q5SETLP - Change Current Large Page Limit	8-79	Tape Files	9-48
Q5SNDMCE - Send Message to Controllee	8-80	Q5GETFIT - Get FIT Field Values	9-50
Q5SNDMCR - Send Message to Controller	8-82	Q5GETN - Read Partition Tape Files	9-55
Q5SNDMDF - Send Message to Dayfile	8-84	Q5GETP - Read Partial Partition Tape Files	9-57
Q5SNDMJC - Send Message to Job Controller	8-87	Q5GIVE - Give File Ownership	9-60
Q5SNDMJS - Send Message to Job Session	8-89	Q5LABEL - Request File from Multifile Set	9-63
Q5SNDMOP - Send Message to Operator	8-90	Multifile Sets	9-67
Q5SNDSTR - Start Controllee Execution	8-92	Q5MAPIN - Map In Virtual Space	9-70
Q5TERM - Terminate Task	8-93	Q5MAPOUT - Map Out Virtual Space	9-72
Q5TERMCE - Disconnect Controllee	8-94	Q5OPEN - Open File	9-74
Q5TIME - Get System Time	8-95	Access Modes	9-74
Q5VRACC - Change Accounting Rate	8-96	Shared Access	9-74
		I/O Buffers	9-82.1
		Implicit I/O	9-83
		Files Connected to Terminals	9-83
		Tape Files	9-83
		Q5PATACH - Attach Pool	9-85
		Q5PCREAT - Create Pool	9-86
		Q5PDESTR - Destroy Pool	9-87
		Q5PDTACH - Detach Pool	9-88
		Q5PERMIT - Change Access Permission Set	9-89
		Tape Access	9-89
		Q5PGRACC - Grant Access to Pool	9-91
		Q5POOLS - List Pools	9-92
		Q5PREACC - Remove Access to Pool	9-93
		Q5PURGE - Purge File	9-94
		Q5PUSERL - List Users With Access to Pool	9-96
		Q5PUTB - Put a Buffer Record	9-97
		Q5PUTN - Write Partition Tape Files	9-98
		Q5PUTP - Write Partial Partition Tape Files	9-100
		Q5READ - Read Block	9-102
		Reading Tape Data	9-104
		Reading PRUs	9-104
		Reading LRUs	9-105
		LRU Description Array	9-105
		Writing Additional Tape Volume Labels	9-105
		Q5REDUCE - Reduce File Space	9-106
		Q5REELSW - Write Additional Tape Volume Labels	9-107
		Q5RETFIT - Return FIT	9-109
		Q5RETURN - Return File Tape Files	9-110
		Q5REWIND - Rewind File Tape Files	9-111
		Q5ROUTE - Route File	9-112
		Q5RQUEST - Request Local File	9-115
		Files Connected to Terminals	9-115
		Tape Files	9-115
9. SYSTEM INTERFACE LANGUAGE (I/O CALLS)	9-1		
SIL I/O Overview	9-3		
Preparing a File for I/O	9-3		
FIT Processing	9-3		
Opening a File	9-4		
Explicit I/O	9-4		
Explicit I/O by Logical Partition	9-4		
Explicit I/O By Physical Block	9-7		
Appending Data	9-7		
Implicit I/O	9-8		
Example of Implicit I/O	9-8		
SIL I/O Calls	9-9		
Q5ATTACH - Attach Permanent File	9-10		
Q5CHANGE - Change File Attributes	9-12		
Q5CHECK - Check I/O Request Status	9-16		
Tape I/O Requests	9-17		
Q5CHECKB - Check Block I/O Request Status	9-19		
Q5CLIOER - Clear Tape I/O Error	9-21		
Q5CLOSE - Close File	9-22		
Tape Label Processing	9-23		
Q5DEFINE - Define Permanent File	9-25		
Q5ENDPAR - Write Partition Delimiter	9-29		
Tape Partition Delimiters	9-30		
Q5GENFIT - Generate FIT	9-31		
Tape File FITs	9-36		
Accessing the Tapes Table Entry	9-37		
Q5GETB - Get a Buffer Record	9-39		
Q5GETFIL - Open or Create and Open a File	9-40		
Mass Storage Files	9-47		

Q5SETFIT - Set FIT Field Values	9-122	Return Buffer	10-19
Q5SKIP - Skip Partition	9-126	Special Characters	10-26
Positioning a Sequential			
Access File	9-127		
Tape Files	9-129		
Q5WRITE - Write Block	9-130	11. VSOS SCREEN SUPPORT ROUTINES	11-1
I/O Buffers Used	9-130	Q9SCR	11-1
Wait Processing	9-131	Definitions	11-2
Appending Blocks	9-132	Synopsis	11-3
Tape Files	9-132	Calling Conventions	11-4
Writing Additional Tape		Memory Use	11-7
Volume Labels	9-132	Descriptions of Individual	
		Routines	11-7
		Limits	11-15
10. COMMON EXECUTE LINE SUPPORTING		Q9SPRINT	11-16
ROUTINES	10-1	Usage	11-16
		Examples	11-18
Conventions	10-1	Alternate Interfaces	11-19
Supporting Routines	10-5	Miscellaneous Definitions and	
Q7ENVIRN	10-5	Guidelines	11-19
Q7MODE	10-6	Format of Display Tables	11-19
Q7PROMPT	10-7	Guidelines for Implementing Table	
Q7KEYWRD	10-8	Displays	11-20
lhs Table	10-12	Guidelines for Implementing Display	
rhs Table	10-15	Tasks	11-20

APPENDIXES

A. CHARACTER SET	A-1	CYBER 170 Arithmetic Conversion	
		Routines	E-5
		Conversion Processing	E-5
B. MESSAGES	B-1	Call Format	E-6
		CYBER 170 to CYBER 200 Integer	
		Conversion	E-6
C. GLOSSARY	C-1	CYBER 200 to CYBER 170 Integer	
		Conversion	E-7
		CYBER 170 to CYBER 200 Floating	
		Point Conversion	E-7
		CYBER 200 to CYBER 170 Floating	
		Point Conversion	E-7
E. FORTRAN DATA CONVERSION ROUTINES	E-1	CYBER 170 to CYBER 200 Numeric	
		Data Transfer Example	E-8
IBM Arithmetic Conversion Routines	E-1	CYBER 170 to CYBER 200	
IBM to CYBER 200 64-Bit Floating		Transfer	E-8
Point Conversion	E-1	CYBER 200 to CYBER 170	
IBM to CYBER 200 32-Bit Floating		Transfer	E-9
Point Conversion	E-2		
CYBER 200 to IBM 64-Bit Floating			
Point Conversion	E-3	F. TAPE LABELS AND FORMATS	F-1
CYBER 200 to IBM 32-Bit Floating			
Point Conversion	E-3	G. DISPLAY PROGRAM EXAMPLE	G-1

INDEX

FIGURES

1-1	CYBER 200 Configuration Example	1-1	4-23.1	LIMITS Control Statement Format	4-60.1
2-1	File Ownership	2-11			
2-2	W Control Word Format	2-20	4-24	LISTAC Control Statement Format	4-62
2-3	L Control Word Format	2-21	4-25	Files Used by the LOAD Utility	4-65
2-4	I Block Control Word Format	2-24	4-26	Controllee File Format (Code and Data Bases Separate)	4-67
2-5	I Format PRU Terminator	2-34	4-27	Controllee File Format (Data Grouped with Code)	4-68
2-6	SI Format PRU Terminator	2-35	4-28	LOAD Control Statement Format	4-74.1
3-1	Task Mapping	3-2	4-29	Example of Interactive LOAD Execution	4-79
3-2	LOGIN Command Format	3-6	4-30	LOADPF Control Statement Format	4-80
3-3	Interactive Execute Line Format	3-11	4-31	LOADPF Output Example	4-87
3-4	Example Batch Input File as Read by the Batch Processor	3-19	4-32	MFLINK Control Statement Format	4-88.1
3-5	USER Control Statement Format	3-22.1	4-33	MFQUEUE Control Statement Format	4-91
3-6	MFGIVE Control Statement Format	3-25	4-34	NORERUN Control Statement Format	4-93
3-7	MFTAKE Control Statement Format	3-25	4-35	NORERUN/RERUN Example	4-93
3-8	Interrupt Subroutine Header	3-29	4-36	OLE Control Statement Format	4-95
4-1	IF Control Statement Format	4-6.1	4-37	PACCESS Control Statement Format	4-97
4-1.1	ELSE Control Statement Format	4-6.2	4-38	PASSWORD Control Statement Format	4-98
4-1.2	ENDIF Control Statement Format	4-6.2	4-39	PATTACH Control Statement Format	4-99
4-1.3	PROC Control Statement Format	4-6.3	4-40	PCREATE Control Statement Format	4-100
4-2	BEGIN Control Statement Format	4-7	4-41	PDELETE Control Statement Format	4-101
4-3	ATTACH Control Statement Format	4-13	4-42	PDESTROY Control Statement Format	4-102
4-4	AUDIT Control Statement Format	4-15	4-43	PDETACH Control Statement Format	4-102
4-5	AUDIT Output Example	4-20	4-44	PERMIT Control Statement Format	4-104
4-6	AUDIT Output Example (if either the ACCOUNT or the MPN parameters are specified)	4-20	4-45	PFILES Control Statement Format	4-105
4-7	CHARGE Control Statement Format	4-22	4-46	PFILES Sample Output	4-105
4-8	COMMENT and Control Statement Format	4-22	4-46.1	PROCEED Control Statement Format	4-106
4-9	COMPARE Control Statement Format	4-22	4-47	PURGE Control Statement Format	4-106.1
4-10	COPY Control Statement Format	4-23	4-48	Q Control Statement Format	4-108
4-11	COPYL Control Statement Format	4-25	4-49	REQUEST Control Statement Format	4-112
4-12	DAYFILE Control Statement Format	4-28	4-50	RERUN Control Statement Format	4-120
4-13	DEFINE Control Statement Format	4-30.1	4-51	RESOURCE Control Statement Format	4-121
4-13.1	DIVERT Control Statement Format	4-31	4-52	RETURN Control Statement Format	4-124
4-14	DMAP Control Statement Format	4-34			
4-14.1	DROP Control Statement Format	4-34.2			
4-15	DUMPF Control Statement Format	4-36.1			
4-16	Directory/Dumped File Format	4-37			
4-17	DUMPF Output Example	4-41			
4-18	EDITPUB Control Statement Format	4-45			
4-19	EXIT Control Statement Format	4-46			
4-20	FILES Control Statement Format	4-48			
4-21	FILES Sample Output	4-48			
4-22	GIVE Control Statement Format	4-50			
4-23	LABEL Control Statement Format	4-51			
		4-53			
		4-55			

4-53	REWIND Control Statement Format	4-126	8-10	Q5DMPACT Call Format	8-34
4-54	SET Control Statement Format	4-127	8-11	Q5ENAMI Call Format	8-35
4-55	SKIP Control Statement Format	4-128.1	8-12	Q5ENATI Call Format	8-37
4-56	SLGEN Control Statement Format	4-129	8-13	Q5GETACT Call Format	8-38
4-57	SLGEN Directive Formats	4-130	8-14	Q5GETCTS Call Format	8-40
4-58	SUBMIT Control Statement Format	4-132	8-15	Q5GETIIP Call Format	8-41
4-59	SUMMARY Control Statement Format	4-134	8-16	Q5GETIRF Call Format	8-42
4-60	SWITCH Control Statement Format	4-137	8-17	Q5GETLP Call Format	8-43
4-61	TASKATT Control Statement Format	4-139	8-18	Q5GETMCE Call Format	8-44
4-62	TV Control Statement Format	4-140	8-19	Q5GETMCR Call Format	8-45
4-63	USER Control Statement Format	4-142	8-20	Q5GETMOP Call Format	8-47
5-1	Typical UPDATE Creation Run	5-2	8-21	Q5GETMPG Call Format	8-48
5-2	Typical UPDATE Correction Run	5-3	8-22	Q5GETPFI Call Format	8-49
5-3	Card Identifier Expansion	5-11	8-23	Q5GETTL Call Format	8-50
5-4	ADDFILE Directive Format	5-15	8-24	Q5GETTN Call Format	8-51
5-5	BEFORE Directive Format	5-16	8-25	Q5GETUID Call Format	8-53
5-6	CALL Directive Format	5-16	8-26	Q5INIT Call Format	8-55
5-7	COMDECK Directive Format	5-17	8-27	Q5INITCH Call Format	8-56
5-8	COMPILE Directive Format	5-18	8-28	Q5LFIHIR Call Format	8-58
5-9	DECK Directive Format	5-19	8-29	Q5LFIPOL Call Format	8-60
5-9.1	DEFINE Directive Format	5-20	8-30	Q5LFIPRI Call Format	8-63
5-10	DELETE Directive Format	5-20	8-31	Q5LFIPUB Call Format	8-66
5-10.1	ENDIF Directive Format	5-21	8-32	Q5LSTBUT Call Format	8-68
5-11	IDENT Directive Format	5-21	8-33	Q5LSTCH Call Format	8-69
5-11.1	IF Directive Format	5-22	8-34	Q5LSTSTB Call Format	8-71
5-12	INSERT Directive Format	5-21.1	8-35	Q5LSTTCB Call Format	8-72
5-13	MOVE Directive Format	5-21.1	8-36	Q5MEMORY Call Format	8-73
5-13.1	PULLMOD Directive Format	5-22	8-37	Q5RECALL Call Format	8-74
5-14	PURDECK Directive Format	5-22.1	8-38	Q5REPREV Call Format	8-75
5-15	PURGE Directive Format	5-23	8-39	Q5RFI Call Format	8-77
5-16	READ Directive Format	5-24	8-40	Q5RUNBIF Call Format	8-78
5-16.1	WIDTH Directive Format	5-24	8-41	Q5SETLP Call Format	8-79
5-17	YANK Directive Format	5-25	8-42	Q5SNDMCE Call Format	8-80
5-18	YANKDECK Directive Format	5-25	8-43	Q5SNDMCR Call Format	8-82
5-19	/ Comment Directive Format	5-26	8-44	Q5SNDMDF Call Format	8-85
5-20	UPDATE Control Statement Format	5-26.1	8-45	Q5SNDMJC Call Format	8-87
6-1	DEBUG Control Statement Format	6-2.1	8-46	Q5SNDMJS Call Format	8-89
6-2	DEBUG Directives	6-3	8-47	Q5SNDMOP Call Format	8-90
6-3	LOOK Control Statement Format	6-13	8-48	Q5SNDSTR Call Format	8-92
6-4	LOOK Directives	6-15	8-49	Q5TERM Call Format	8-93
6-5	DUMP Control Statement Format	6-24	8-50	Q5TERMCE Call Format	8-94
7-1	CHKPNT Subroutine Format	7-1	8-51	Q5TIME Call Format	8-95
7-2	CHKPNT Error Codes	7-3	8-52	Q5VRACC Call Format	8-96
7-3	Return Word Format	7-4	9-1	Q5ATTACH Call Format	9-10
8-1	Q5ADVISE Call Format	8-8	9-2	Q5CHANGE Call Format	9-12
8-2	Q5CPUTIM Call Format	8-10	9-3	Q5CHECK Call Format	9-16
8-3	Q5DCDDST Call Format	8-11	9-4	LRU Description Format	9-18
8-4	Q5DCDMSC Call Format	8-13	9-5	Q5CHECKB Call Format	9-19
8-5	Q5DCDPFI Call Format	8-18	9-6	Q5CLIOER Call Format	9-21
8-6	Q5DCDPLB Call Format	8-29	9-7	Q5CLOSE Call Format	9-22.1
8-7	Q5DESBIF Call Format	8-31	9-8	Q5DEFINE Call Format	9-25
8-8	Q5DISAMI Call Format	8-32	9-9	Q5ENDPAR Call Format	9-29
8-9	Q5DISATI Call Format	8-33	9-10	Q5GENFIT Call Format	9-31
			9-11	Tapes Table Entry Format	9-37
			9-12	Q5GETB Call Format	9-39
			9-13	Q5GETFIL Call Format	9-40
			9-14	Q5GETFIT Call Format	9-50
			9-15	Q5GETN Call Format	9-56
			9-16	Q5GETP Call Format	9-57

9-17	Q5GIVE Call Format	9-60	10-9	lhs Table Entry Format	10-14
9-18	Q5LABEL Call Format	9-63	10-10	rhs Table Format	10-15
9-19	Q5MAPIN Call Format	9-70	10-11	rhs Table Entry Format (First Word)	10-16
9-20	Q5MAPOUT Call Format	9-72			
9-21	Q5OPEN Call Format	9-75	10-12	rhs Table Entry Format, Type 2	10-17
9-22	Q5PATACH Call Format	9-85	10-13	rhs Table Entry Format, Type 3	10-17
9-23	Q5PCREAT Call Format	9-86	10-14	rhs Table Entry Format, Type 4/6	10-18
9-24	Q5PDESTR Call Format	9-87			
9-25	Q5PDTACH Call Format	9-88	10-15	Return Buffer Format	10-20
9-26	Q5PERMIT Call Format	9-89	10-16	Return Buffer Entry Format (First Word)	10-20
9-27	Q5PGRACC Call Format	9-91			
9-28	Pool List Entry Format	9-92	10-17	Return Buffer Entry Format, Types 1 and 2	10-21
9-29	Q5POOLS Call Format	9-92			
9-30	Q5PREACC Call Format	9-93	10-18	Return Buffer Entry Format, Type 3	10-21
9-31	Q5PURGE Call Format	9-94			
9-32	Q5PUSERL Call Format	9-96	10-19	Return Buffer Entry Format, Type 4	10-22
9-33	Q5PUTB Call Format	9-97			
9-34	Q5PUTN Call Format	9-98	10-20	Return Buffer Entry Format, Type 5	10-23
9-35	Q5PUTP Call Format	9-100			
9-36	Q5READ Call Format	9-102	10-21	Return Buffer Entry Format, Type 6	10-23
9-37	Q5REDUCE Call Format	9-106			
9-38	Q5REELSW Call Format	9-107	10-22	Return Buffer Entry Format, Type 7 with Zeroed Flags	10-24
9-39	Q5RETFIT Call Format	9-109			
9-40	Q5RETURN Call Format	9-110	10-23	Return Buffer Entry Format, Type 7 with Set Flags	10-24
9-41	Q5REWIND Call Format	9-111			
9-42	Q5ROUTE Call Format	9-112	10-24	Return Buffer Entry Format, Type 8 with One Set Flag	10-25
9-43	Q5RQUEST Call Format	9-116			
9-44	Q5SETFIT Call Format	9-122	10-25	Return Buffer Entry Format, Type 8 with Two Set Flags	10-26
9-45	Q5SKIP Call Format	9-126			
9-46	Q5WRITE Call Format	9-130	11-1	Symbols Known LOOKUP	11-5
10-1	Key-Dependent Parameter Format	10-3	11-2	Q95SCR Maxima and Minima	11-15
10-2	Q7ENVIRN Call Statement Format	10-5	D-1	FIT Format	D-3
10-3	Q7MODE Call Statement Format	10-6	F-1	ANSI Standard Tape Label Groupings	F-3
10-4	Q7PROMPT Call Statement Format	10-7			
10-5	Q7KEYWRD Call Statement Format	10-10	F-2	Unlabeled Tape Files	F-4
10-6	lhs Table Pointer Configuration	10-11	F-3	Summary of Tape Blocks Per Group	F-11
10-7	lhs Table Format	10-12	F-4	Single Volume Tapes	F-13
10-8	lhs Table Header Format	10-13	F-5	Multivolume Tapes	F-14

TABLES

2-1	Concurrent File Access Modes	2-16	4-6	Interaction of USER and POOL Parameters for LOADPF	4-86.1
2-2	Blocking Type, Tape Format, and Record Type Combinations	2-33	4-7	Recovery Error Codes	4-88
3-1	Program States	3-8	4-8	Logical Structure Conversion	4-90
3-2	Logical Structure Conversion	3-26	4-9	Input Queue Status Identifiers	4-109
4-1	Control Statements	4-1	4-10	Task Status Identifiers	4-110
4-2	Interaction of USER and POOL Parameters for AUDIT, DUMPF, and LOADPF	4-17	4-11	Output File Status Identifiers	4-111
4-3	GIVE Default Access Permission Sets	4-54	5-1	Summary of UPDATE Call Parameters	5-4
4-4	Access Permission Sets Listed	4-61	5-2	Summary of UPDATE Directives	5-5
4-5	Results of Listing and Controllee File Searches	4-66	5-3	File Contents and Update Mode	5-19
			8-1	SIL Non-I/O Calls	8-2
			9-1	SIL I/O Calls	9-1
			9-2	Calling Parameter Value Ranges	9-9

9-3	Blocking Type, Tape Format, and Record Type Combinations	9-36	A-2	Hexadecimal-Octal Conversion	A-3
9-4	Q5GIVE Default Access Permission Sets	9-62	A-3	Hexadecimal-Decimal Conversion	A-4
10-1	Execute Line Special Characters	10-27	B-1	Diagnostic Messages	B-2
11-1	Symbols Known by LOOKUP	11-5	B-2	System Utility Error Messages	B-26
11-2		11-14	B-3	System Error Codes	B-104
A-1	American National Standard Code for Information Interchange (ASCII) With Punched Card Codes and EBCDIC Translation	A-2	B-4	Tape Error Codes	B-106
			D-1	File Information Table	D-2
			F-1	Required ANSI Label Formats	F-5
			F-2	Optional Label Formats	F-8
			F-3	Tape Group Separators	F-10

The virtual storage operating system (VSOS) controls a CYBER 200 Computer System. This chapter gives a general description of CYBER 200 hardware and an overview of VSOS.

SYSTEM CONFIGURATION

A CYBER 200 system configuration consists of the CYBER 200 mainframe and peripheral system components. Peripheral system components include a maintenance control unit (MCU), on-line mass storage, on-line tapes, and one or more front-end computer systems.

The system configuration must also include hardware for communication between the CYBER 200 mainframe and the peripheral system components. Communication within a CYBER 200 system is performed by a loosely coupled network (LCN).

Figure 1-1 shows one possible CYBER 200 system configuration.

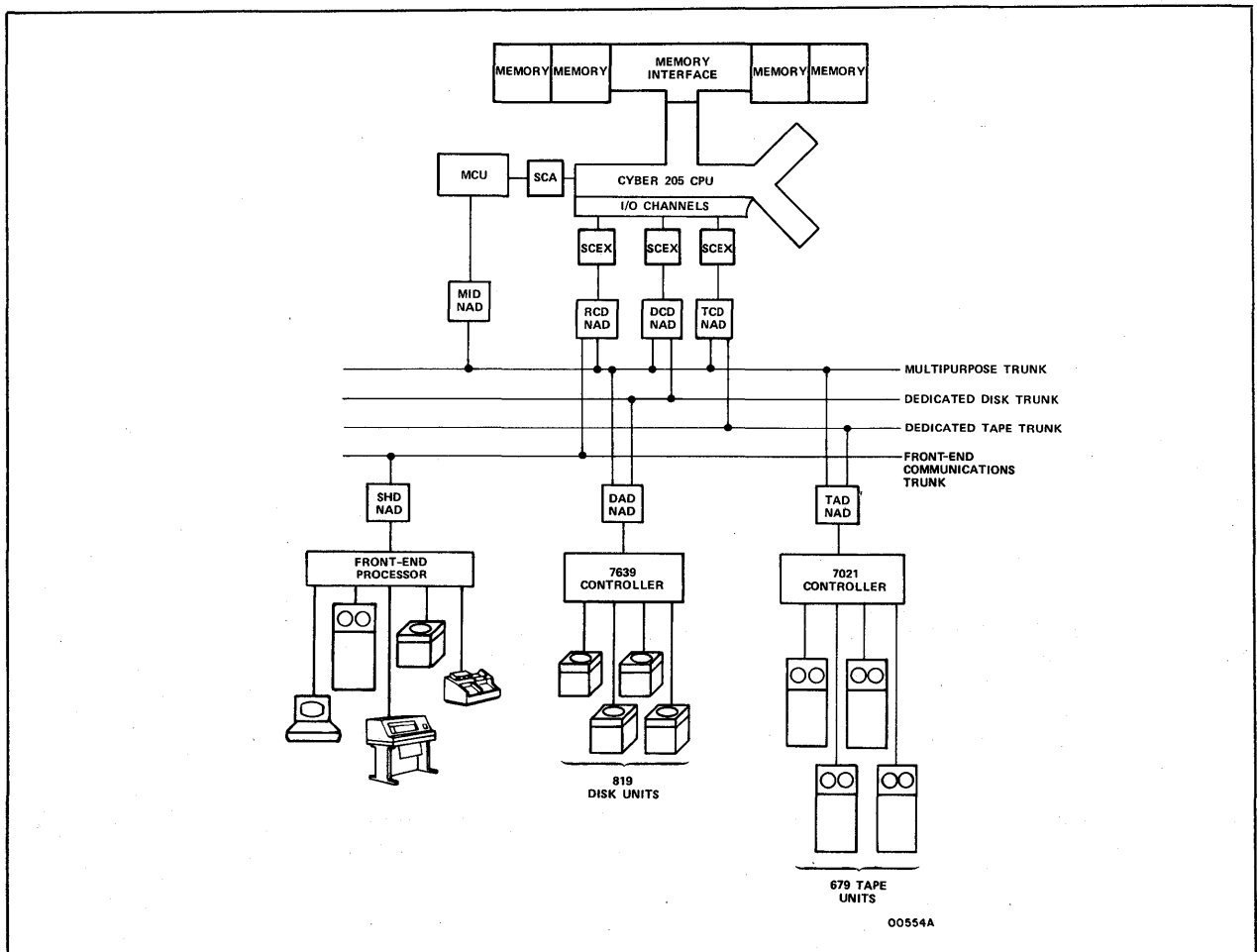


Figure 1-1. CYBER 200 Configuration Example

The components of a CYBER 200 system are as follows:

- CYBER 200 mainframe
- CYBER 200 memory
- Maintenance and control unit (MCU)
- Loosely coupled network (LCN)

CYBER 200 MAINFRAME

A CYBER 200 system configuration is designed for effective use of the CYBER 200 mainframe. Each CYBER 200 mainframe contains a vector processor, a scalar processor, I/O channels, and central memory. The CYBER 200 Model 205 can have up to 16 million words of central memory and up to 16 I/O channels.

The vector processor performs vector instructions (instructions that use streams of operands to produce streams of results). The scalar processor performs scalar instructions (instructions that produce one result) and directs vector processing and central memory data transfers. The I/O channels control data communication between the scalar processor and the LCN.

CYBER 200 MEMORY

The CYBER 200 is a virtual memory machine. The program space is limited only by the available disk space.

Program space (or virtual space) is the range of virtual addresses used by the execution of a program (a task). When a task is executed, its virtual space is mapped to physical space (central memory and disk space). During execution, the task is assigned central memory only for the code and data it is currently using. The rest of the task code and data remains on disk. When the task requires additional code or data that is currently not in memory, VSOS automatically copies it from disk into central memory.

The process of copying code and data in and out of central memory is called paging or implicit I/O. The units transferred in and out are called pages. VSOS uses two page sizes, a small page and a large page. The small page size is selected during VSOS autoloading. It can be one, four, or sixteen 512-word blocks. The large page size is always 128 512-word blocks (65536 words).

Paging requires high-speed data communication between a CYBER 200 I/O channel and a 7639 disk controller connected to 819 disk units. Disk communication for the CYBER 200 Model 205 is performed by the LCN.

MAINTENANCE AND CONTROL UNIT

Operation control and diagnostic functions for the CYBER 200 mainframe are performed at the MCU. The CYBER 18 MCU is described in the VSOS Version 2 Operator's Guide.

LOOSELY COUPLED NETWORK (LCN)

The LCN is a communications network consisting of network access devices (NADs) connected by trunk lines. Each NAD can connect to as many as four trunk lines; each trunk line can connect to as many as 24 NADs. A NAD can communicate with another NAD if both are connected to the same trunk line.

The function of a NAD within the LCN depends on the system component to which it is connected. A DCD NAD connected to a CYBER 200 disk I/O channel communicates with a NAD connected to a 7639 disk controller. A NAD connected to a CYBER 200 tape I/O channel communicates with a NAD connected to an Advanced Tape System (ATS) controller. A NAD connected to a CYBER 200 Remote Host Facility (RHF) I/O channel communicates with a NAD connected to a front-end computer system. The NAD connected to the MCU may communicate with all NADs in the LCN.

OPERATING SYSTEM

The CYBER 200 operating system has the following three components:

- Resident system
- Virtual system
- Privileged system tasks

RESIDENT SYSTEM

The resident system runs in monitor mode; it is always resident in main memory. It references memory by physical addresses, rather than virtual addresses. When the CPU is in monitor mode, interrupts are inhibited, and some additional instructions are enabled.

The resident central operating system has the following four primary parts:

- KERNEL - responsible for processor management and message handling
- PAGER - responsible for memory management and page swapping
- XIOCALL - responsible for explicit I/O
- NPSCALL - responsible for tape I/O

All access interrupts (page faults), as well as certain messages dealing with memory allocation, are passed to PAGER by KERNEL. PAGER dynamically allocates both large and small pages and performs all required implicit input/output necessary to free memory pages and obtain the pages causing access interrupts.

PAGER determines dynamically which pages of a user's virtual address space have the most activity. These pages are the working set of pages for the task at that time.

VIRTUAL SYSTEM

The virtual system is a task that runs in job mode and references memory by virtual address. It communicates with the resident system via system messages and can modify system tables. The virtual system performs resource allocation, file management, and message processing functions.

PRIVILEGED SYSTEM TASKS

Privileged system tasks executed for special user numbers. These special user numbers can issue privileged and nonprivileged system calls and have additional privileges. Unlike virtual system tasks, privileged system tasks with the exception of the Input/Queue Manager (IQM) and the Interactive Transfer Facility Servicer (ITFS) cannot modify system tables directly.

Privileged system tasks perform some of the work of the virtual system. This results in a reduction of virtual system overhead and frees the virtual system to process other functions. Work such as operator communication and the handling of job input and output is currently done by privileged system tasks.

VSOS USER INTERFACE

The VSOS user interface includes the communications software that allows access to a CYBER 200 system, system utilities, and the system interface language (SIL).

VSOS communications software is the CYBER 200 Remote Host Facility (CYBER 200 RHF).

A system utility is a system-supplied program for performing a common user function. Utility execution is initiated by execution of its control statement within a job. Usually, a utility is directed by control statement parameters and/or utility directives. The VSOS utilities are described in chapters 4, 5, and 6 of this manual.

SIL is a set of subroutines that user programs can call to perform system functions. Chapter 8 describes SIL procedures that perform non-I/O functions; chapter 9 describes SIL procedures that perform I/O functions.

REMOTE HOST FACILITY (RHF)

The RHF is the software that communicates with the LCN. Each computer system attached to the LCN executes its own RHF software.

The RHF software that executes on a CYBER 200 system is called CYBER 200 RHF. CYBER 200 RHF consists of application programs and a command driver routine. Each application program (with the exception of MFLINK) executes as privileged system task. An application program exists for each RHF function, including sending queue files, receiving queue files, sending permanent files, receiving permanent files, and receiving interactive requests.

The CYBER 200 RHF user interface is described in chapter 3 of this manual. The CYBER 200 RHF operator interface is described in the VSOS 2 Operator's Guide.

CYBER 200 COMPARISON

Significant differences exist between the hardware and software of a CYBER 200 system and that of a CYBER 170 front-end system. Some of the differences that affect application programs are as follows:

- CYBER 200 memory words are 64 bits.
- CYBER 200 uses a hexadecimal, rather than an octal, number system. The hexadecimal number sequence is: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
- CYBER 200 character data is a subset of 8-bit ASCII, not 6-bit display code. Appendix A contains character code tables.
- The CYBER 200 is a virtual memory machine. CYBER 200 programs do not require overlays or segmentation. An efficient program uses its data in such a way as to minimize paging for data access. It also uses the appropriate page size for its data and program (large or small pages).

VIRTUAL MEMORY ADDRESSING

The execution of a controllee is called a task. Each task has its own virtual space--the range of virtual addresses that it uses. The CYBER 200 system translates virtual addresses to physical addresses without affecting the task.

The system page table associates each physical page in memory with a task and a virtual address range in that task's virtual space. The page table entry also indicates whether the page can be written or can only be read.

A successful association between a virtual address and an entry in the page table causes that entry to be moved to the head of the table; all other entries are moved down by one place.

The first 16 entries in the page table are kept in high-speed registers; the registers are examined in parallel with a simultaneous associative compare. An unsuccessful compare results in a sequential search through the remainder of the table held in main memory. Table entries for infrequently used pages automatically float to the end of the table.

If an address has no entry in the page table, the task requesting the address is interrupted. Normally, the system provides the space addressed by requesting the page moved to central memory from the disk. The program continues processing from the point of interruption.

REGISTER FILE

The CPU contains a file of 256 64-bit registers used for instruction and operand addressing, indexing, and field length counts, and as a source or destination for register-type instructions. The register file is accessible to assembly language programs and to FORTRAN language programs that use special calls.

By convention, its contents are divided into several areas that can be used to pass parameters to another routine, access data for programs, trace execution, and hold constants. The contents of the register file are dumped to an output file as part of abnormal job termination; a similar dump can be obtained during program execution through the debugging facilities available in the system.

Register file conventions are described in appendix D of volume 2 of this manual.

A file is a collection of data accessible to you and to the system by name. File names are from one to eight characters long. User-created file names must begin with a letter and be composed of letters and digits; however, they must not begin with the characters Q5 through Q9. System-created file names can begin with any character.

VSOS is a file-based operating system. Knowledge of file concepts is important for the understanding of system function and facilities. Not all general concepts apply to all of the device types supported by the system. This chapter covers file concepts in general and then relates each to mass storage, magnetic tape, and interactive terminal files.

FILE ATTRIBUTES

VSOS identifies attributes of each file when it is created. These attributes may be explicitly selected by you (as parameters on the REQUEST statement, for example) or by the system (default values). Attributes are maintained by the operating system for the life of the file. They can be permanently changed (via a SWITCH or TASKATT statement) if the owner chooses to do so (file ownership is described later in this chapter). Some can also be overridden temporarily within a single task (via a SIL subroutine call); for example, a task can treat a file as a bit string by temporarily using a record type of "U".

For more information on file attributes, refer to Logical File Structures later in this chapter.

FILE TYPES

VSOS recognizes the following file types:

- Controllee files
- Data files
- Drop files
- Output files

The file type determines the way the system uses and disposes of the file.

CONTROLLEE FILES

A controllee file (also called a virtual file or a virtual code file) is an executable file. It is generated by the LOAD utility (refer to LOAD - Generate Controllee File in chapter 4 of this manual). A task is the execution of a controllee file (refer to Initiating Controllee Execution in chapter 3).

A controllee file contains all information needed for execution of the object code contained within the file. Its first 512-word block is called its minus page.

The minus page contains control information used by the operating system.

A detailed explanation of the minus page and other system tables is included in the VSOS 2 Reference Manual, Volume 2.

DATA FILES

A data file (also called a physical file) is any CYBER 200 file that is not a controllee file or a drop file. A data file is not executable.

For example, an object code file is a data file. Although an object code file is not executable, it is used as input for generation of an executable controllee file.

A data file does not have a minus page.

DROP FILES

The system creates a drop file for each task it executes. When a task references virtual address space that is not associated with the controllee file or with another mass storage file, the system associates the virtual address space with the task drop file.

If the task modifies a page of its controllee file and the page must be swapped out, it is copied to the task drop file and not to the controllee file. Later, if the page is referenced again by the task, the modified page from the drop file is paged in, but the unmodified page from the controllee file is not.

Modified pages of a mass storage file opened for implicit I/O can be written to the drop file instead of to the original file (refer to Q5OPEN in chapter 9 of this manual).

A task can create its own drop file if no blocks have yet been written to its drop file. The system-created drop file for the task is then destroyed. Refer to volume 2 of this manual for more information.

The system creates the drop file with read and execute access permissions. (File access permissions are described later in this chapter.)

If you do not specify the drop file length on the LOAD or TASKATT statement, the system determines an appropriate drop file length for the task, using the controllee file and common block lengths. The minimum drop file size is an installation parameter (system release value, #25 blocks).

Drop File Naming Convention

The system derives the name of a drop file from the name of the controllee file. The controllee name is shifted right one character, truncating the rightmost character (unless there were fewer than eight characters), and one digit is added as the leftmost character of the drop file name. The digit added corresponds to the controllee level of the task, as follows:

- If the task is the batch processor, the digit is 1.
- If the task is initiated by the batch processor or at an interactive terminal, the digit is 2.

The maximum controllee level is 9.

For more information, refer to Controllee Chains in chapter 3 of this manual.

Restarting a Task

If a task has terminated because of a time limit condition or entry of the break character, restart the task by executing its drop file. (The break character is an installation parameter; the release value is !.)

NOTES

- At a security-sensitive site that has imposed the safeguards described in chapter 7 of the Installation Handbook, the dump file of a task that has time-limit aborted may not be restartable until enabled by the site security administrator.
- An attempt to restart a task by executing its drop file fails if the task has terminated abnormally or the operator has dropped the task.

Drop file execution fails if VSOS is using a different small page size than it used when the drop file was created.

After abnormal task termination, the task drop file exists as a local file. To save the task drop file, the job must make the drop file permanent by defining it. For example, suppose a controllee file named GO, initiated at an interactive terminal, aborts because it has exceeded its time limit. According to the drop file naming convention, the name of the task drop file is 2GO. To restart the task, enter the following execute line:

2GO.

Bound Explicit and Implicit Maps

In the drop file, the system maintains tables that associate the addresses of program data on mass storage with the virtual addresses used by the program.

When a file is opened for explicit I/O, the system enters the information in a table called the bound explicit map. The information in the bound explicit map is used by the system routines that process explicit I/O requests.

When a file is opened for implicit I/O and then mapped to a program array, the mapping information is entered in a table called the bound implicit map. Entries in the bound implicit map associate a range of virtual addresses with mass storage space. An entry exists for each virtual address range used.

The size of the drop file map tables is limited. An attempt to add an entry to a full drop file map is a fatal error. To avoid the error, a program that uses numerous virtual address ranges should create a file, open it for implicit I/O, and map the virtual address ranges into the file.

OUTPUT FILES

Output files contain data to be processed by an output device. When the file is closed or the task that created the output file terminates normally, VSOS generates an output file with a valid disposition code. This file is sent to the appropriate privileged user task for processing to the output device. After the file is processed, it is destroyed.

Output files with disposition codes not recognized by the front-end system are evicted, and an informative message is sent to the user.

By default, an output file is returned to the front-end system that submitted the job or to the interactive terminal that initiated the task. An output file is assigned the default disposition code and internal and external characteristics unless specified otherwise.

For RHF output files, specify output parameters on the statement that submits the job file or on an MFQUEUE statement.

Print Files

Each output file generated by a task within a job is saved as a member of a print family (refer to Job Processing in chapter 3 of this manual). The print family is not processed until the job dayfile (PXXfamm) is added to the print family. In addition, the print family of an RHF job must end with a last group file (PYYfamm), which contains routing information.

VSOS processes a print family before it is routed to the front-end system. It generates a compressed ASCII file with ANSI carriage control characters as follows:

1. VSOS expands compressed blanks on the file. [Blank compression is described under Record Mark Delimited (R) Record Format in this section.]
2. VSOS reads the internal characteristics (IC) field of the file's File Index entry. If its IC is ASCII with ANSI carriage control, VSOS assumes that the carriage control characters are correct and performs no conversion.

If its IC is ASCII with ASCII carriage control, VSOS converts the ASCII carriage control to ANSI carriage control. The ASCII form feed control character, FF (#0C), when it appears as the first character of the file after a unit separator, US (#1F), is replaced by the ANSI page eject control character 1 (#31). If there is no form feed control character, an ANSI page eject control character is added. The ASCII single space, which is a line without the FF after the US, is changed by inserting the ANSI space-one-line control character, blank (#20), after the US.

3. The files in a print family are linked, forming one file. Each file delimiter character, FS (#1C), except the last is removed.

Print Control Characters

Print file control characters follow either ASCII or ANSI control character conventions. In the ASCII schema, print control is governed completely by the appearance of ASCII control characters. The FF control character must be the first character of a file or must immediately follow a unit separator (US). The ASCII control characters and their effect on vertical spacing are as follows:

<u>Control Character</u>	<u>Vertical Spacing</u>
FF (#0C)	Page eject
US (#1F)	Single space

In the ANSI print control conventions, the first character of a printer/display output record is not printed or displayed; it is interpreted for vertical spacing control. The first character of each output record directed to the card punch or to any device other than a printer or a display unit is transmitted and recorded just as any other character in the record is, without any special action. Characters and their effects on vertical spacing before the printing or displaying of the next record are as follows:

<u>Character</u>	<u>Hexadecimal Code</u>	<u>Vertical Spacing</u>
blank	#20	Single space.
0	#30	Double space.
1	#31	Page eject.
+	#2B	No vertical advances; move to the first position of the same line.
-	#2D	Triple space.
other	other	Single space.

OUTPUT FILE ERROR PROCESSING

When the system attempts to send the output-file-family to the appropriate output device, a condition may occur that may temporarily block the output from arriving at the output device. You may view the status of the output by executing a Q,0 command interactively (refer to chapter 4 of this manual, Control Statements). Further details of the temporary error condition are contained in the last-group-file of the output-file-family. The last-group-file contains information needed for routing the file to its destination. It begins with the prefix Q5L or PYY, depending on whether the output is from the MFQUEUE or is the output from a batch job. You can retrieve the output-file-family by asking the operator to GIVE the output to its originating user number or to the system user number (refer to the VSOS Version 2 Operator's Guide, chapter 4). You can then attach the last-group-file and examine its contents, using the LOOK utility. Details of the error condition are written at hex word address A0 and begin with a 16-character header as follows:

```
RETRIES = xxxxxx
```

xxxxxx is the retry count. A circular error message buffer follows next. The last five temporary error messages are saved. The circular error message begins at hex word address 18A. Each entry is 112 characters long and has the following format:

```
hh.mm.ss MESSAGE FROM xxx FOLLOWS: yyy...yyy
```

hh.mm.ss is the time of the error encounter;

xxx is either RHF, SIL, or a remote host PID;

yyy...yyy indicates a maximum of an 84-character error message sent by xxx.

If the output-file-family is permanently blocked from arriving at the output device, and your job (either batch or interactive) is still active, the message is sent to your job, informing you that the file has been evicted and telling you why the file was blocked. If your job is no longer active, the informative message is sent to the origin of the job.

FILE DURATION

A file can be categorized by the duration of its existence in the system. Categories of user files are scratch, local, and permanent. Scratch files are destroyed at the end of the task that uses them. Local files are destroyed at the job's end. Permanent files are stored until their owners explicitly destroy them (file ownership is described later in this chapter).

SCRATCH FILES

A scratch file is a system-created file that VSOS uses while executing a task. VSOS destroys all scratch files when the task terminates normally. If the task terminates abnormally, the system saves the files as local files so that it can provide you with information on the cause of the abnormal termination.

LOCAL FILES

A local file is a temporary file created and used only for the duration of the interactive session or batch job. Unless you define the file as a permanent file, the system destroys it at the end of the batch job or interactive session that created it.

VSOS utilities that write local files discard the local file you have specified prior to writing into the file. This is done because many utilities need special file characteristics when writing into a file, and allowing the utility to create its own files relieves you of remembering that sort of detail. However, if you define the file prior to execution of the utility, the utility does not discard the file.

A local file exists only for the user number and batch job or interactive session that created the file.

The name of a local file must be unique among both the local files and the attached private permanent files currently assigned to the job or interactive session.

PERMANENT FILES

A permanent file exists until you explicitly destroy it. It remains stored (on mass storage) after the termination of the job that created it.

Each permanent file is described in an entry on the disk pack on which it resides. During VSOS autoloading, the pack file entry is copied to a system file; it is later copied to a system table. Attempts to attach the permanent file search the system table, not the disk packs themselves.

The system table can hold only 256 permanent file entries per user number. Therefore, even if you own more than 256 permanent files, only the first 256 files whose entries VSOS finds and copies are accessible to you. An inaccessible file could become accessible if the following occurs:

- You, as the owner of the inaccessible file, purge one or more accessible permanent files.
- During a subsequent VSOS autoloading, the entry for the previously inaccessible file is copied to the system file.

A permanent file must be attached before its data can be accessed.

FILE USAGE CONTROLS

File usage is controlled by the following means:

- File security levels
- File ownership
- File access permissions

Additional file usage controls that can be imposed at security-sensitive sites are described in chapter 7 of the Installation Handbook.

FILE SECURITY LEVELS

VSOS validates eight security levels, from level 1, the lowest security, to level 8, the highest security. The default security level, if none is specified, is determined by an installation parameter value. The system released value for default security level is 1.

Each VSOS user number has a maximum security level assigned to it. No job or task belonging to the user number can execute at a security level higher than the maximum level for the user number.

The security level of a job or interactive session determines the maximum security level of the files it references or creates. The following are ways of specifying a security level.

<u>Security Level Associated with</u>	<u>Assigned by</u>
Interactive terminal session	LOGIN line
Batch job submitted via RHF	USER statement
File	Statement or program call that creates the file

A task cannot attach or purge a file whose security level is higher than that of the task. A task cannot give a file to a user number whose maximum security level is lower than that of the file. Similarly, a task cannot give a file to a pool whose pool boss has a maximum security level lower than that of the file (refer to Pool Files in this chapter).

FILE PATTERNING

Both permanent and local files are patterned when they are purged if the installation parameter IP_PLEV value is greater than or equal to the security level of the file. The released value of IP_PLEV is 4.

Local files are patterned either when you explicitly return the file or when the system does it automatically at the end of a job.

FILE OWNERSHIP

Each file has an owner. It can be owned by a user number, a pool, or the public file list. The corresponding file ownership categories are private, pool, and public. The file ownership category of a file determines certain access characteristics of the file.

A local file is a private file. A permanent file can be private, pool, or public.

Ownership rights of nonprivileged users depend on the file ownership category of the file. A privileged user has ownership rights over all files except local files belonging to other users.

A file name must be unique among the files in its ownership category currently accessed by a job. A private file could have the same name as a pool file or a public file attached to a job. However, a private permanent file attached to a job or interactive session cannot have the same name as a private local file currently belonging to the job or interactive session.

Your job can have a local file with the same name as one of your permanent files, as long as the permanent file is not attached by that job. Parallel jobs in execution for the same user number can have local files of the same name.

Figure 2-1 illustrates the use of file management utilities for each file ownership category. The utilities named in figure 2-1 are described in chapter 4 of this manual.

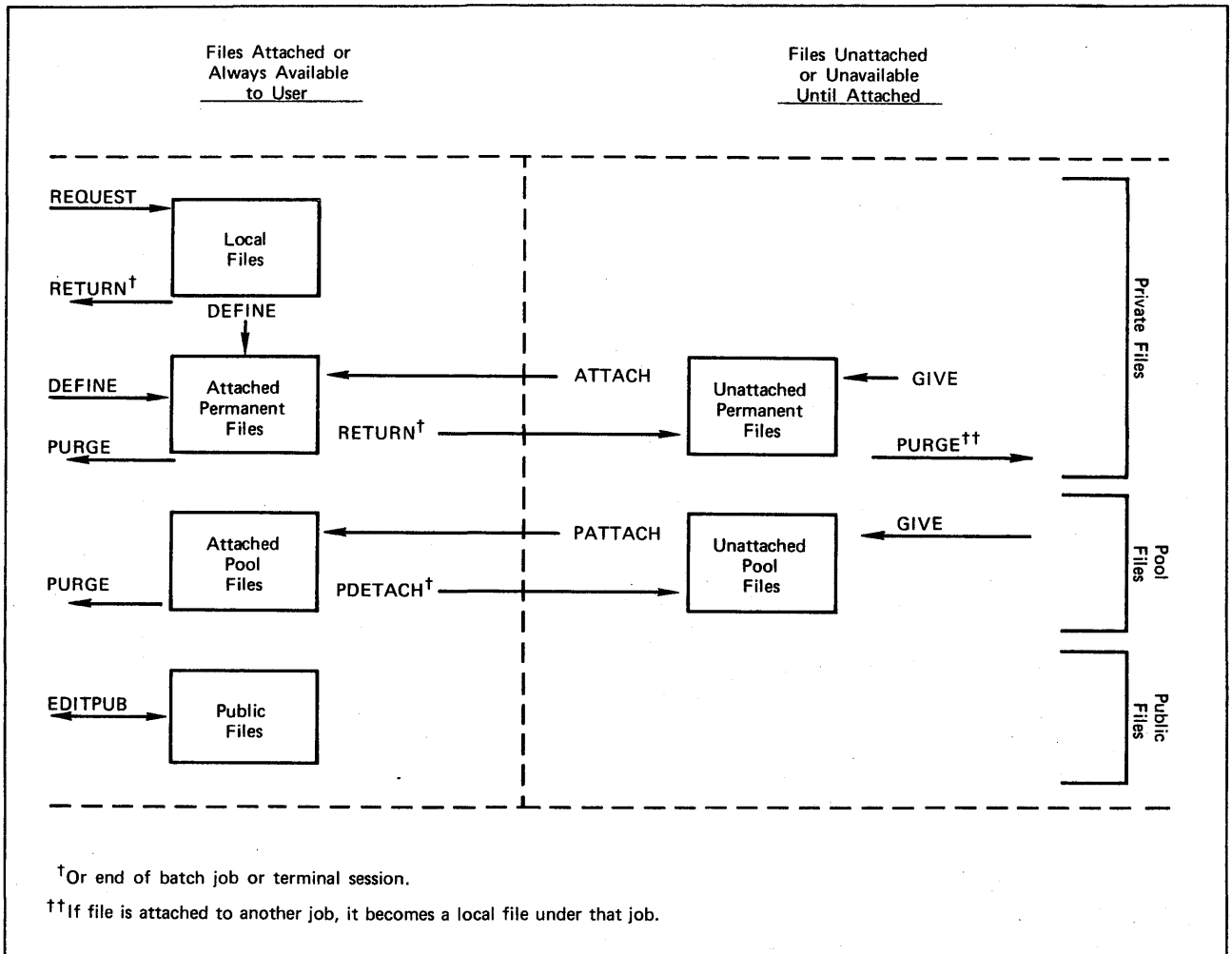


Figure 2-1. File Ownership

File Search Hierarchy

When searching for a file, VSOS searches the files attached to the job or interactive session by file ownership category. The categories are searched in the following order:

1. Private files (local and attached permanent).
2. Pool files, in reverse order of that in which their pools were attached. If attached, the system pool is always the last pool searched.
3. Public files.

Private Files

A private file is owned by one user number. The owner of a private file is the user number that created the file or the user number given file ownership by the previous file owner. Only the file owner can access a private local file.

The file owner and privileged users can access a private permanent file; other users can access the file if the owner defines one or more access permission sets for the file. (Refer to File Access Permissions in this chapter.)

For a task to access a private permanent file, the file must be attached to the job or interactive session. A new private permanent file is attached when it is defined; an existing private permanent file must be explicitly attached. The file remains attached until it is returned or the job or session terminates.

Only the file owner can change private file characteristics and ownership. Private file ownership changes when the file owner gives the file to another user number or to a pool. A privileged owner can give the file to the public file list.

Pool Files

A pool file is a file owned by a pool. A pool is a mechanism for sharing VSOS files. (The preferred method of file sharing is via explicit file access permissions as described later, not via pools.) A pool or pool file acts much like a file catalog. Each pool has an entry in the pool list. Its name must be unique within the pool list. A pool name must consist of from one to eight characters, beginning with a letter.

The user who creates a pool by adding a name to a pool list is the pool boss for that pool. Only the pool boss can perform the following functions.

- Give files belonging to the pool to another user
- Purge files belonging to the pool
- Grant other users access to the pool
- Remove user authorization to access the pool
- Destroy the pool

A pool member is a user granted access to a pool.

Any user can give files to any pool; pool membership is not required. If, however, the user gives a file to a pool of which he or she is not a pool member, access to the file is forfeited because the file now belongs to the pool.

When a pool member attaches a pool, all files belonging to the pool (except files having a security level greater than the security level of the job or interactive session) are attached.

A job can have up to four pools attached at one time (including the system pool if one exists). Pool attaches and detaches are relative to a job rather than to a user number. When a batch job terminates, it detaches all pools it attached. Pools attached by an interactive task are detached at logout.

When a job is initiated by a user and one or more pools is already attached to the job, VSOS sends the following message to the job dayfile or to the interactive terminal.

WARNING**ATTACHED POOLS

An interactive user can enter \$P to list the attached pools.

System Pool

A system pool is a pool of system files automatically attached when the user logs in or submits a batch job.

All characteristics of a system pool are the same as those of other pools except its rank in the file search hierarchy. A file is searched for in the system pool after all other attached pools have been searched.

Use of a system pool can be selected during VSOS autoloading. If system pool use is selected, VSOS searches the system pool for a file before it searches the public files. This means that if a system pool file has the same name as a public file, the system pool file effectively replaces the public file.

Public files instead of system pool files can be used when the system pool is explicitly detached.

Public Files

The system attaches all public files to a batch job or interactive session when the job or session is initiated. The job or session can then use any public file.

Public files usually contain assemblers, compilers, and other general-purpose routines. VSOS stores the utilities described in this manual as executable public files.

All public files belong to user number 000000, indicating system ownership. The site administrator or another privileged user determines the files owned by user number 000000.

FILE ACCESS PERMISSIONS

An access permission to a file grants you a mode of access to that file. The access permissions are read, write, append, modify, and execute.

Each access permission allows only that mode of access. No access permission implicitly grants another access permission.

Before processing an access request, VSOS checks the access permissions applicable to you and to the file. This determines whether the request is valid.

To open a file for explicit I/O, read, write, append, or modify permission to the file is required (refer to Explicit I/O in this chapter).

To open a file for implicit I/O, read permission to the file is required. To write to a file using implicit I/O, write and read permission are required (refer to Implicit I/O in this chapter).

Read Permission

Read permission to a file allows you to read data from the file, to reposition the file by skipping records or blocks, and to route the file.

With read permission, a file for explicit I/O or implicit I/O can be opened.

Write Permission

Write permission to a file allows you to write data on the file, overwrite existing file space, and extend the file space.

With write permission, a file for explicit I/O can be opened. Read permission is required to open a file for implicit I/O; write permission is required to write to the file after it is opened.

Append Permission

Append permission to a file allows you to write data on the file, but only at the end of the existing file data. Append access is valid for sequential access files only.

Append permission allows a mass storage file for explicit I/O to be opened. It does not allow a file for implicit I/O or a file assigned to a device other than mass storage to be opened.

To append data to an empty file (a new file that contains no data), only append permission is required; the file is positioned for appending data when it is opened.

To append data to a file containing data, both read permission and append permission are required. Read permission is required to position the file at the end of its existing data. After opening the file, position the file after its data; if the file ends with a file delimiter, the file must be positioned before the delimiter so that the new data overwrites the delimiter. For more information on positioning the file, refer to Appending Data in chapter 9 of this manual.

Modify Permission

Modify permission to a file allows you to write data to the file if the write operation does not extend the file. With modify permission, any record within the file or any block from the beginning of the file through the highest block previously referenced in the file can be rewritten.

Modify access is valid only for direct access files.

Execute Permission

Execute permission allows you to request execution of the file. The file is executed if it is in executable format; that is, it must be a controllee file generated by the LOAD utility.

If you have not obtained execute permission to a file, VSOS aborts a request to execute the file and sends a message to the job dayfile or to the interactive terminal. (It also sends the message to the system dayfile).

Execute permission does not grant you permission to perform system functions other than execution of the file.

Access Permission Sets

An access permission set is the set of access permissions you are granted to a file. An access permission set can explicitly grant any combination of access permissions.

The possible access permission sets differ for each of the file ownership categories as follows:

- A public file has a general access permission set that applies to all users of the file.
- A pool file has a general access permission set that applies to all pool members. It also has a pool boss access permission set that can grant additional access permissions to the pool boss.
- A private local file has an access permission set applicable to the file owner.
- A private permanent file always has an access permission set applicable to the file owner. It can also have a general access permission set applicable to all users (excluding the file owner) and individual access permission sets for individual user numbers.

If a private permanent file has a general access permission set, every user has at least the access permissions granted by that set. The file owner can grant additional permissions to a user with an individual access permission set.

A file can have individual access permission sets for up to 16 users.

NOTE

For each file for which individual access permission sets are specified, the maximum number of files the owner can have (256) is reduced by one file.

Concurrent File Access

Private permanent files can be shared with other jobs. Private local files cannot be shared.

A private permanent file cannot be shared between users initially. Sharing becomes possible when the owner of the file defines one or more access permission sets for other users (general or individual access permission sets).

A shared file is one whose read access and execute access are available to more than one job at a time. (The job can belong to the file owner or to another user.) If a job is accessing a file in read mode and/or execute mode, another job can access the file in read mode and/or execute mode, but cannot access the file in write, append, or modify mode.

If a job is accessing a file in write, append, or modify mode, no other job can access the file until the first job terminates access.

Concurrent file access is summarized in table 2-1.

Table 2-1. Concurrent File Access Modes

If a job is accessing the file in:	Another job requests access in:				
	Read Mode	Execute Mode	Write Mode	Append Mode	Modify Mode
Read Mode	Allowed	Allowed	Denied	Denied	Denied
Execute Mode	Allowed	Allowed	Denied	Denied	Denied
Write Mode	Denied	Denied	Denied	Denied	Denied
Append Mode	Denied	Denied	Denied	Denied	Denied
Modify Mode	Denied	Denied	Denied	Denied	Denied

FILE I/O

VSOS provides the following two means of reading and writing data:

- Explicit I/O
- Implicit I/O

Both types of I/O can read and write virtual files and physical files.

EXPLICIT I/O

Explicit I/O uses buffers within the program space. An explicit I/O request causes one or more blocks of data to be copied to or from program buffers.

The system can transfer data to a buffer while the program continues executing. When the program needs the data from an explicit read request or needs to write more data after an explicit write request, it must check to see whether the request has been completed. If the request has not been completed, the program must wait until completion before using the data (for read) or the buffer (for write).

Explicit I/O is most efficiently used for transferring multiple blocks of data using multiple buffers.

IMPLICIT I/O

Implicit I/O does not use intermediate buffers. A task does not issue implicit I/O requests. VSOS performs an implicit read operation when a task references data that is not currently in central memory; it performs an implicit write operation when the file is closed or the system reassigns the physical central memory space.

To read or write data implicitly, a program opens the file for implicit I/O and then calls the SIL Q5MAPIN subroutine to map a mass storage file to an array in its virtual space. When the task references an element in the mapped-in array, VSOS ensures that the data mapped to that element is available in memory. When the file is closed, the data stored in the array is copied to the mass storage file.

If a program array is not mapped to a file, VSOS maps the array to space in the drop file (refer to Drop Files in this chapter for more information).

Implicit I/O is most efficiently used for accessing only one block of data at a time or for accessing the same file area repeatedly.

LOGICAL FILE STRUCTURES

The following three levels of file structures are supported by SIL:

- Records
- Groups
- Files

A record is the smallest unit of associated data managed by SIL. Typically, a record is the result of a write statement in an application program.

The next higher level of file structure is a group of records. A file that contains group indicators is called a structured file. Multiple groups may appear in a file. Not all record formats support group structures. An example of a multigroup file is a VSOS job deck with one input file in it; the control statements comprise the first group, and the input file is the second group.

The highest level of structure is the file itself. It is known to the operating system by its file name. It is a single entity to SIL, and programs cannot read beyond the end of a file.

LOGICAL RECORD FORMATS

A logical record format is the means by which SIL subroutines recognize a logical record structure. A logical record structure enables you to read and write logical records (and, if supported, logical groups).

You may specify the record format used for a file you create. SIL supports the following six record formats (not all record types are supported for all device types):

- ANSI fixed length (F)
- Record mark delimited (R)
- Undefined structure (U)
- Control word delimited (W)
- CYBER Record Manager (L) control word
- System block (B)

ANSI Fixed Length (F) Record Format

SIL writes a fixed number of bytes for each F format record. F format does not support group delimiters.

You must specify the fixed record length (RLMAX) before reading or writing F format records. SIL is unable to determine the length of a record if RLMAX is not specified.

When reading F format records, SIL compares the RLMAX and working storage area length (WSL) values. It returns an error message if the WSL value is less than that of RLMAX.

When writing F format records, if RLMAX is greater than the length of data to be written as a record (the working storage area length), SIL appends padding characters to the record. If the data length is greater than RLMAX, SIL writes a fixed length record and discards the excess data. SIL error messages are returned for both of these conditions.

Unless a padding character is specified, SIL uses the default character specified by an installation parameter [system release value, blank character (ASCII code #20)].

Record Mark Delimited (R) Record Format

The R record format is the released system default record format.

When writing R format records, SIL terminates each record with the record mark character. You may specify a record mark character or use the default character specified by an installation parameter [system release value, ASCII US (#1F)].

The R format supports group delimiters only if the record mark character is ASCII US or RS (#1E). If the record mark character is ASCII US or RS, SIL recognizes the ASCII GS (#1D) character as the group delimiter. If RS is the record mark character, US characters are considered data. The ASCII FS (#1C) is the file delimiter for R format files.

When reading R format records, SIL searches for the record mark character delimiting the record. If it does not find the character, SIL reads the number of characters specified by the WSL value, skips to the beginning of the next record, and returns an error message.

When writing R format records, SIL compresses consecutive blanks unless instructed otherwise. It replaces strings of more than two blanks with two character codes, ASCII ESC character (#1B) followed by the number of blanks plus #30. SIL adds #30 to the number so that the value cannot be mistaken for another ASCII control character code.

When reading R format records, SIL expands compressed blanks unless instructed otherwise.

Undefined Structure (U) Record Format

A U record format file has no structure; it does not support any delimiters.

SIL considers the file as a continuous byte string. You specify the number of bytes that SIL reads or writes for each call. The data length specified can vary with each call.

Control Word Delimited (W) Record Format

The W record format uses control words as record, group, and file delimiters. The control word format is shown in figure 2-2.

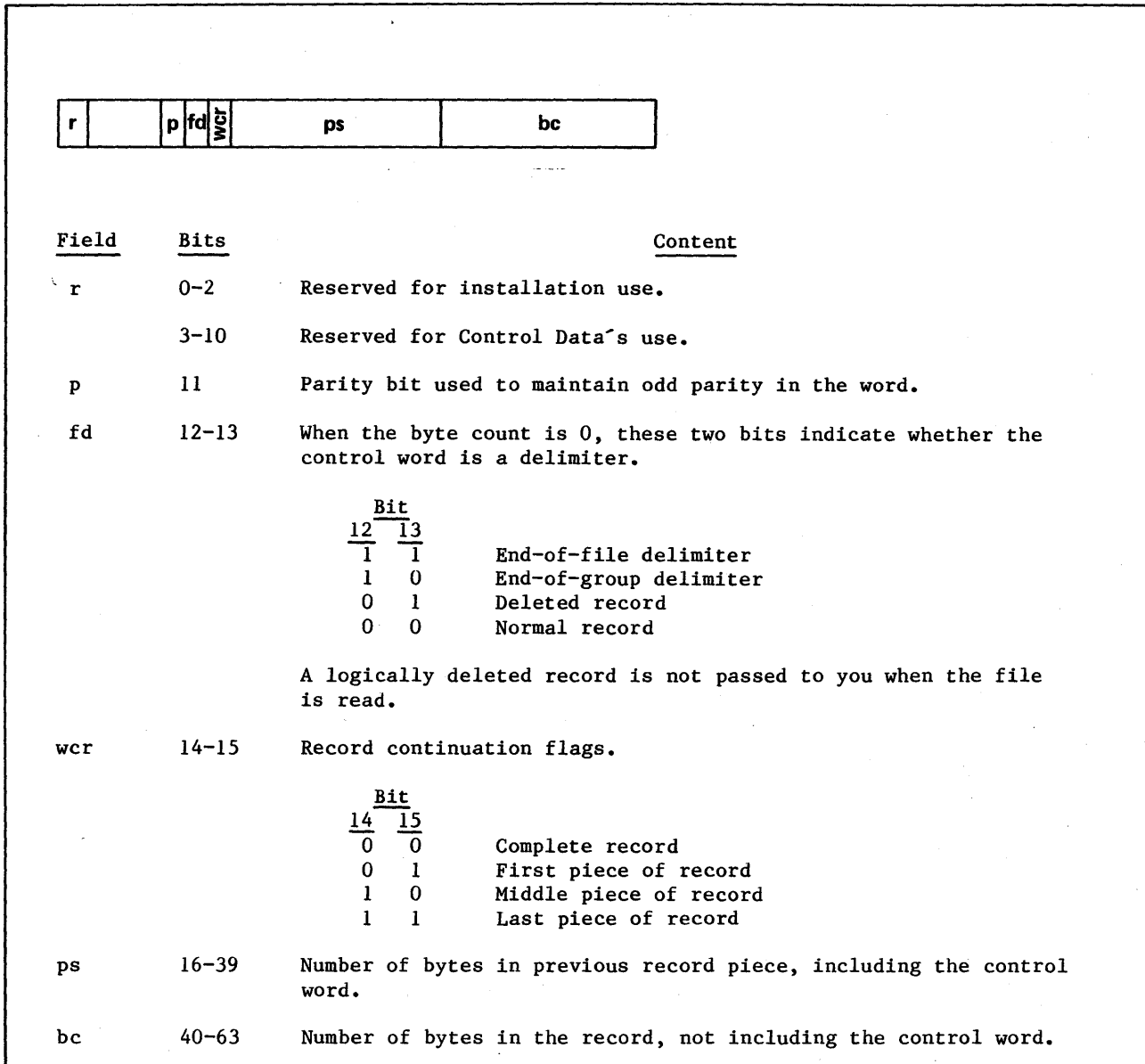


Figure 2-2. W Control Word Format

SIL can write a W format record in more than one piece. It prefixes each piece of a record with a control word describing the piece. The maximum size of a piece is $2^{24}-1$ bytes.

The group and file delimiters are control words following the last record in the group or file. They are distinguished by a flag indicating the partition level of the control word.

SIL prefixes each W format record it writes with a control word. The control word contains the number of bytes in the record and the number of bytes between the control word and the beginning of the previous control word.

When reading a W format record, SIL reads the control word and then transfers the number of bytes of data specified in the control word byte count field. It does not transfer the control word to the working storage area.

When reading a W format file at the group level, SIL transfers data (including record control words) until it reads a group control word. When reading at the file level, SIL transfers data (including group and record control words) until it reads a file control word.

When writing a W format record, SIL writes the control word before writing the record data.

When a group or file is written, the first word in the working storage area must be a record control word. You can include other control words within the working storage area to delimit records within the group or delimit groups within the file.

When writing at group or file levels, SIL writes the data and then writes the group or file control word. It enters zero in the byte count field of the control word and the number of bytes to the beginning of the previous control word in the previous size field.

CYBER Record Manager Control Word (L) Record Format

The L record format allows VSOS to interchange tapes with a CYBER 170 system. By specifying L record format, VSOS can read tapes written by a CYBER 170 system, using the CYBER Record Manager (CRM) control word (W) record format. Similarly, the CYBER 170 system can read tapes written by VSOS, using the L record format.

Each L record is an integral number of data words, prefixed by a control word. Figure 2-3 shows the L control word format.

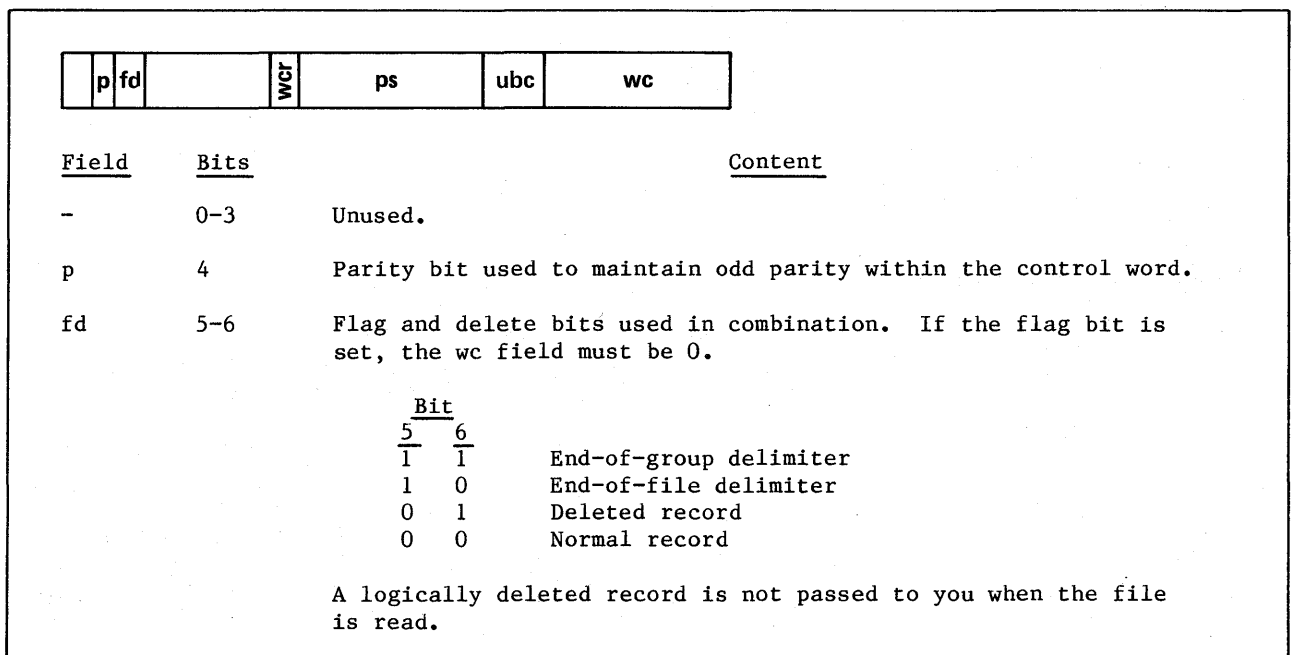


Figure 2-3. L Control Word Format (Sheet 1 of 2)

<u>Field</u>	<u>Bits</u>	<u>Content</u>																		
-	7-19	Unused.																		
wcr	20-21	Record continuation flags.																		
		<table style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="2"><u>Bit</u></th> <th></th> </tr> <tr> <th><u>20</u></th> <th><u>21</u></th> <th></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Complete record</td> </tr> <tr> <td>0</td> <td>1</td> <td>First piece of record</td> </tr> <tr> <td>1</td> <td>0</td> <td>Middle piece of record</td> </tr> <tr> <td>1</td> <td>1</td> <td>Last piece of record</td> </tr> </tbody> </table>	<u>Bit</u>			<u>20</u>	<u>21</u>		0	0	Complete record	0	1	First piece of record	1	0	Middle piece of record	1	1	Last piece of record
<u>Bit</u>																				
<u>20</u>	<u>21</u>																			
0	0	Complete record																		
0	1	First piece of record																		
1	0	Middle piece of record																		
1	1	Last piece of record																		
ps	22-39	Number of words in the previous record, including the control word. If the current record is the first record, this field is 0.																		
ubc	40-45	Number of rightmost bits not used in the last word (integer from 0 through 59).																		
wc	46-63	Number of words in the record piece, not including the control word.																		

Figure 2-3. L Control Word Format (Sheet 2 of 2)

As shown in figure 2-3, the control word contains the size of the previous record and the size of the current record. VSOS uses this information for two purposes: to position the file by records and as a data reliability check. VSOS checks to ensure that the number of words read for a record matches the number of words written, as recorded in its control word.

The VSOS L control word is a 60-bit CYBER Record Manager W control word, right justified with zero fill, in a 64-bit CYBER 200 word. The conversion between the two control words is performed by the VSOS assembly/disassembly option (ADO), which automatically converts between 60-bit and 64-bit word sizes. Specifying the L record format also specifies the ADO.

The L record type is valid with I blocking only. VSOS I blocking is interchangeable with CYBER Record Manager I blocking (refer to I Blocking in this chapter).

System Block (B) Record Format

A system block (B) record is the one VSOS logical record type that is directly related to the physical layout of the data. A record is equivalent to a logical record unit (LRU). LRUs are defined by the tape format used (refer to Tape Formats in this chapter).

The primary use of the B record format is to read tapes not written on a CDC system (stranger tapes).

Reading a stranger tape requires that the tape data be read block by block. Because a B record is equivalent to an LRU and, for V format, an LRU is equivalent to a tape block, the specification of B record type and V tape format means that each Q5GETN call reads one tape block.

BLOCKING TYPES

Blocking refers to the physical layout of the data on the device. The basic unit or block is called a physical record unit (PRU). On a disk, a sector is a PRU. On a tape, the data delimited by interblock gaps (IBGs) is a PRU. A block can contain more than one logical record; records can span blocks.

VSOS supports the following three blocking types:

- Character count (C)
- Internal (I)
- Record count (K)

C Blocking

Character count (C) blocking writes a fixed number of characters per block. The block size is the maximum physical record unit (MPRU) size. The MPRU size is determined by the device and format.

For I, SI, and LB tape formats and for disk, the MPRU size is fixed. For V tape format, you may specify the MPRU size.

The last block in a tape file can be smaller than the MPRU size.

I Blocking

Internal (I) blocking is used for CDC CYBER 170 tape interchange using the L record type. VSOS I blocking is equivalent to CYBER Record Manager I blocking.

For I and SI tape formats, the PRU size is fixed. For V tape format, you may specify the MPRU size.

Each I block is an integral number of words, prefixed by a control word. Figure 2-4 shows the I-block control word format.

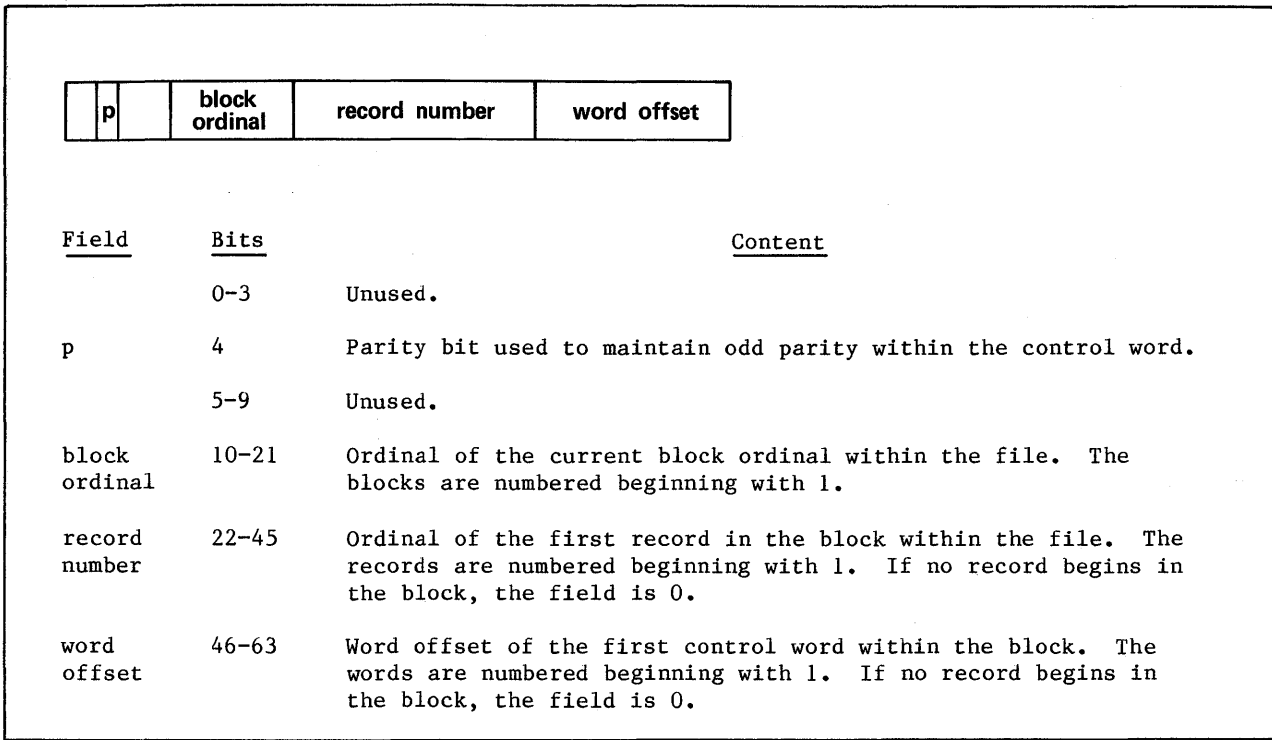


Figure 2-4. I Block Control Word Format

The VSOS I-block control word is a 60-bit CYBER NOS Record Manager I block control word, right justified with zero fill, in a 64-bit CYBER 200 word. The conversion between the two control words is performed by the VSOS ADO, which automatically converts between 60-bit and 64-bit word sizes. Specifying I blocking also specifies the ADO.

I blocking is valid with the L record type only. The VSOS L record type is interchangeable with the CYBER Record Manager W record type (refer to CYBER Record Manager Control Word (L) Record Format in this chapter).

K Blocking

K blocking writes a fixed number of records per block. Records cannot span blocks. The records per block (RPB) value must be specified.

K blocking is valid only for variable (V) tape format because if the record length is variable, K blocks are also variable in length.

The last block in the tape file can have fewer records than the RPB value.

FILE ORGANIZATION

VSOS supports both sequential and direct access file organizations. Sequential access is the default organization.

Sequential Access Organization

Sequential access to a file accesses each record in sequence. An explicit I/O call to a sequential access file reads or writes data at the current file position.

Sequential access files can use any record format. Sequential access file organization is valid for mass storage files, tape files, and files connected to a terminal.

Direct Access Organization

Direct access to a file accesses a record by its record number. Records in a direct access file are numbered consecutively, starting with 1.

Direct access file organization is valid only for mass storage files with F format records.

Because each record in a direct access file has a fixed length, SIL can compute the byte address of the beginning of a record as follows:

$$(\text{record number} - 1) * (\text{fixed record length in bytes})$$

Unless the site changes the maximum record length installation parameter, specify a maximum record length when creating a direct access file. The released default value for the maximum record length is zero. A maximum record length of zero prevents writing on the direct access file, since zero is used as a special value by SIL, meaning unlimited.

When the record number is omitted on explicit I/O calls, a task can read or write direct access records sequentially without changing the organization attribute of the file. SIL sets the current record number to zero when it opens or rewinds the file; it increments the current record number by one for each record read or written.

DEVICE CHARACTERISTICS

VSOS supports the following three device types:

- Mass storage
- Magnetic tape
- Interactive terminal

MASS STORAGE FILES

Mass storage (disk) is essential to system operation on a virtual memory machine. Each page of central memory must be allocated the corresponding space on disk. Data is automatically paged in and out between central memory and disk, as required by the executing tasks. As a result, most of the concepts described earlier in this chapter apply to mass storage files.

All four file types--controllee, data, output, and drop files--are supported on mass storage. The three file duration types, scratch, local, and permanent, are supported on mass storage. Security level is a maintained attribute for mass storage files, and patterning is done on the disk, depending on the value of this attribute. All three ownership categories are supported (private, pool, and public); only mass storage files can belong to pools or be public files. All five access permissions apply to these files. Only mass storage files can be shared among users. Both explicit and implicit I/O can be done to and from mass storage files. Record and blocking types are as follows:

<u>Block Type</u>	<u>Record Types</u>
C	F
C	R
C	U
C	W

All disk blocks are 512 words. Sequential and direct access file organizations are supported.

Typical job control statements for creating mass storage files are REQUEST and DEFINE, specifying file attributes required by the user. DEFINE can create a new permanent file or can make a local file permanent. File attributes are set for a local file at creation time; parameters specified on the DEFINE statement are then ignored.

File Space Allocation

A mass storage file can comprise an indefinite number of segments. A segment is a contiguous area of disk space; all segments do not necessarily reside on the same disk.

Disk space is allocated automatically, as needed by a file. Physical mass storage devices are treated by the system as a logical unit, called a device set. Files overflow from one set member to another within the device set. Every device set has a set number of two hexadecimal digits. The set number concatenated to the string DVST forms the set name DVSTnn. The set name is stored in the label of the disk. If your private file has overflowed, one or more segments may not be available because a device is down. The truncated parameter must be used to access partial information.

Initial File Space Allocation

When creating a file, specify the following space allocation conditions:

- Initial file length
- Whether the file can be segmented
- Whether the file is extendable
- Device set to include the file

By default, a file is one device allocation unit (DAU) in length; it may be segmented and extended. The DAU is selected by the site administration.

When creating a file that cannot be segmented, VSOS attempts to allocate the file on the pack that is least full and that has sufficient contiguous space. When creating a file that can be segmented, VSOS allocates it on the pack that is least full, in as many segments as necessary. If the entire file cannot fit on the pack, it overflows to the pack with the next most available space in the device set.

Files created for internal use by the operating system are contiguous and nonextendable. All files created by existing programs, utilities, and FORTRAN run-time routines default to extendable files.

Select a device set by specifying any pack in the set using the PACK= parameter on DEFINE, REQUEST, or COPY. Allocation starts with the least full allocatable pack, which is ON. Specifying the PACK= parameter does not necessarily put a file on a specific pack.

File Extensions

If a file is extendable, VSOS extends it for either of the following events:

- The length of a file to be mapped in for implicit I/O is less than the current small page size and has write access permission.
- A task attempts to write data beyond the current end of the file.

Initial File Extension - If no space is available for a file extension, the task is terminated with a message.

When mapping files for implicit I/O, VSOS extends the file to cover the first page in the mapped region, providing that the file is extendable and write access is permitted. If the file is nonextendable with write access permitted, only pages that have allocated disk space can be mapped. If the file has only read access permitted, the mapped region may include the last block of the file but may not cover pages for which no file space is allocated.

Additional File Extensions - When a task that is writing on a file references an address beyond the existing end of the file, VSOS attempts to allocate a new segment if the file is extendable. The size of each extension can be controlled by the user.

The maximum size to which a file can be extended is limited only by an installation parameter. The maximum size is 4 billion characters.

When mapping files that are extendable and have write permission, the mapped region may be arbitrarily large. VSOS extends the file to cover pages as they are accessed.

TAPE FILES

VSOS provides the following two means of accessing tape data:

- A tape can be read on a front-end system and its data copied to a CYBER 200 mass storage file via RHF.
- A tape can be read by an on-line tape drive.

The information in this subsection applies only to on-line tape I/O. For information on transferring a file to or from a front-end system, refer to appendix D in this manual.

VSOS supports all four file types (controllee, data, drop, and output) on tape. However, it supports only explicit I/O for tape; implicit I/O is not supported. Therefore, although a controllee or drop file can be copied to tape, the tape copy cannot be executed or used to restart a task.

Tape files are local private files; they cannot be shared. Read access, write access, or read and write access are allowed. You can request access to tape labels and tape data separately on a REQUEST statement. For example, read access to the tape labels and write access to the tape data can be requested.

Because users often store related files on a single tape volume or set of volumes, VSOS allows you to specify a set of characteristics applicable to all files on the tape volumes. This set of general characteristics is specified on the REQUEST statement for the tapes and is associated with the multifile name (MFN) specified on the statement.

If the tape contains only one file, the file can be referenced by its MFN. However, if the tape contains more than one file, each file is referenced by a LABEL statement that specifies the MFN and a logical file name (LFN) for the file. The LABEL statement can also override any of the characteristics specified on the REQUEST statement; the override applies only to that file.

Tape Drive Reservation

Because tapes are dynamic to VSOS (instead of permanently mounted as 819 disks are), the level of tape resources needed by the system must be identified.

The RESOURCE statement of a batch job that accesses tape files must reserve one or more tape units for use by the job. The initial tape unit reservation count is maintained until the job terminates unless a RETURN statement or Q5RETURN call decrements the reservation count.

An interactive session cannot use the RESOURCE statement. Therefore, tape drives cannot be reserved in interactive sessions. Instead, the request for a tape file reserves a tape drive. If all tape drives are committed, an error message is returned.

Volume Assignment

To read or write data on a tape volume, associate the tape volume with a local file name. To do so, you must request a tape file with a REQUEST control statement or Q5RREQUEST call. The statement or call specifies the file name and its tape volumes.

A tape volume is identified by its volume serial number (VSN). Each tape volume should have its VSN on an external label readable by the operator. If it is an ANSI standard labeled file, it also has its VSN recorded in its VOL1 label. A new tape can be blank labeled by the operations staff with a BLANK control statement so that VSOS can identify the tape.

When VSOS processes a tape file request, it searches a system table for the first tape volume specified on the request. The system table contains an entry for each volume currently mounted on an on-line tape unit. If VSOS finds the requested volume and it is not already assigned to a job, it assigns the tape unit to the file. If VSOS does not find the volume, it prompts the operator to mount the volume. When the volume is mounted, VSOS assigns the tape unit to the file.

If the tape file request does not specify a tape volume for the file, VSOS prompts the operator to mount a tape volume and enter its VSN with a command assigning the volume to the file.

Volume Switching

VSOS supports multivolume tape files. Up to 255 tape volumes can be associated with a file name. The volumes are specified in a VSN list on the tape file request.

By default, VSOS performs automatic tape switching; that is, when the end of the current tape volume is encountered, VSOS switches to the next volume in the file and continues the interrupted operation.

If you like, you can prevent automatic tape switching. To do so, you must specify the ETP parameter on the Q5OPEN call that opens the tape file. Subsequently, when VSOS encounters the end of the current tape volume, it terminates the current operation and returns control to the caller. The caller must then call the Q5REELSW routine to switch the file to the next volume and issue another request to resume the terminated file operation.

A Q5RREQUEST call specifies the order in which the tape volumes in the VSN list are used. The tape volumes are used in sequential order.

Tape Labeling

VSOS can read and write unlabeled, nonstandard labeled, and ANSI standard labeled tape files. You specify the labeling type on the tape file request.

Unlabeled Tape Files

An unlabeled tape file contains no identifying tape labels. When an unlabeled tape volume is mounted, VSOS cannot read the VSN from the tape. The operator mounts the tape and then enters the VSN of an unlabeled tape volume.

When an unlabeled tape file is opened, VSOS positions the tape at its load point. The job can then read or write data on the tape.

After writing data on the tape, VSOS marks the end of the data with an end-of-file indicator. The end-of-file indicator is either an EOF1 label or two tape marks, depending on the tape format. Figure F-2 in this manual illustrates both unlabeled tape formats.

Tape Files with Nonstandard Labels

A tape file with nonstandard labels contains labels that do not conform to ANSI standard X3.27-1978. As with an unlabeled tape file, VSOS cannot identify a nonstandard labeled tape file by a VSN written on the tape. The operator must enter the tape VSN when mounting the tape.

As is the case for a file with ANSI standard labels, write label processing (LPROC=W) can be requested for a nonstandard labeled file. Write label processing overwrites existing header labels when the file is opened. However, the header labels must be specified in an array specified on the Q5OPEN call. End-of-file labels can also be specified in an array on the Q5CLOSE call that closes the file.

VSOS positions the file after its header labels when it opens the file, just as it does for a file with ANSI standard labels. However, VSOS does not verify the contents of the labels, and it cannot position the tape by label groups within the tape data.

Tape Files with ANSI Standard Labels

A tape file with ANSI standard labels contains labels that conform to level 2 of ANSI standard X3.27-1978. The ANSI standard label formats are given in appendix F of this manual.

ANSI standard labels serve to identify and delimit the data on the tape volume and each file on the volume. A multifile set must use ANSI standard labels to delimit the files in the set.

Required Labels - The following ANSI labels are required.

<u>Label</u>	<u>Description</u>
VOL1	Marks the beginning of a tape volume
HDR1	Marks the beginning of a tape file
EOF1	Marks the end of a tape file
EOV1	Marks the end of a tape volume (used only if the end of the volume precedes the end of the file)

A BLANK control statement writes the VOL1 label on a new tape. This function is normally performed by the operator, who must specify the VSN to be recorded in the label. An accessibility character that restricts access to the volume can also be specified. If the VOL1 label contains a nonblank accessibility character, attempts to use the volume must specify the accessibility character.

The HDR1 label identifies the file it precedes. It can also contain an accessibility character to restrict access to the file. As with the VOL1 label, if the HDR1 label contains nonblank accessibility characters, attempts to access the file must specify the accessibility character. Also, if the accessibility character is A, the user attempting to access the file must be the owner of the volume, as identified in the VOL1 label.

The expiration date in the HDR1 label prevents attempts to overwrite the file before the file retention period has expired. You specify a retention period when the file is written, and the retention period is added to the creation date to determine the expiration date for the file.

When accessing a file belonging to a multifile set, specify its file identifier, its file sequence number, or both, as recorded in its HDR1 label. When the file is opened, VSOS searches for the HDR1 label containing the specified values. When it finds the specified HDR1 label, it checks to ensure that the accessibility character and expiration date do not prevent access to the file. If access is not prevented, the tape is positioned at the beginning of the file data. For more information on multifile sets, refer to the LABEL control statement or Q5LABEL call description in chapter 9 of this manual.

Except on V format unlabeled files, the end of the written data is always marked by an EOF1 label. If the end of the volume precedes the end of the file, the end of the data on the volume is marked by an EOVI label.

Both EOF1 and EOVI labels contain a block count. The EOF1 block count is the number of data blocks in the file. The EOVI block count is the number of blocks (data and labels) written on the volume. When a file is opened for read access, VSOS keeps count of the number of blocks read. When it reads an EOF1 or EOVI label, it compares its current block count with the block count in the label. If the values do not match, it returns a dayfile message, notifying the user of the discrepancy.

Optional Labels - The following additional optional labels can also be specified.

<u>Label</u>	<u>Description</u>
UVLn	Sequence of one to nine additional volume labels (n is a digit, 1 through 9).
HDRn	Sequence of one to eight additional header labels (n is a digit, 2 through 9).
UHLa	Sequence of additional header labels (a can be any character).
EOFn	Sequence of one to eight additional end-of-file labels (n is a digit, 2 through 9).
UTLa	Sequence of additional trailer labels (a can be any character).

Except for the label identifier and the label length (80 bytes), VSOS does not check the contents of the optional labels. The optional label formats are shown in appendix G of this manual.

HDRn and UHLa labels can be specified on the Q5OPEN call that opens the file for write access. EOFn and UTLa labels can be specified on the Q5CLOSE call that closes the file. Subsequent Q5OPEN and Q5CLOSE calls to the file can return the contents of the optional labels.

To write the optional beginning-of-volume and end-of-volume labels (UVLn and UTLa), perform end-of-tape processing (refer to Volume Switching in this chapter). Call Q5REELSW to continue file processing. The call can also specify labels to be written at the end of the current volume and labels to be written at the beginning of the next volume.

Tape Data Recording

A tape file request can specify the data recording density and data conversion options. However, if the tape is already labeled, the specified recording density and character set conversion cannot conflict with the density and character set used for the tape labels.

Recording Densities

VSOS can read and write only 9-track tapes. The 9-track tapes can use either of the following two recording densities:

PE 1600 cpi

GE 6250 cpi

Data Conversion Options

A tape file can be a coded or a binary tape file. VSOS considers coded tape data to be a stream of character codes; it considers binary tape data to be a stream of bits. Only character data is stored on coded tape files; binary tape files can store either character or numeric data.

By default, VSOS assumes that a tape file is binary. To indicate that the file is coded, the tape file request must specify the CONVERT parameter.

If the CONVERT parameter is specified, VSOS converts the tape data to and from the character codes in the character set specified by the CM parameter. The CM parameter can specify either the ASCII or the EBCDIC character set. (The ASCII character set is the default.)

The assembly/disassembly option (ADO) provides for binary tape interchange between the CYBER 170 and CYBER 200 systems. The ADO is valid for binary files only; it is not valid for coded files.

If selected, the ADO automatically converts between the CYBER 170 60-bit word size and the CYBER 200 64-bit word size. For reading data, each 60 bits read is right justified with zero fill within a 64-bit CYBER 200 word. For writing data, only the rightmost 60 bits of each 64-bit word are written.

VSOS provides conversion routines to convert 60-bit CYBER 170 numeric data to and from 64-bit CYBER 200 numeric data formats. These routines are described in appendix D of this manual.

Tape Data Organization

Tape data has three levels of organization. Like mass storage data, tape data is organized into logical records and, if supported by the record type, logical groups. Tape data is also organized into tape blocks (PRUs). Depending on the tape format used, tape PRUs are grouped into logical record units (LRUs).

When accessing a tape file, you specify the tape format, blocking type, and record type for the file or use the default values (release values, LB format, C blocking, and R record type).

Not all combinations of tape format, blocking type, and record type are valid; table 2-2 shows the valid combinations.

Table 2-2. Blocking Type, Tape Format, and Record Type Combinations

Blocking Type and Tape Format									
Record Type	BT=C			BT=I			BT=K		
	I,SI	LB	V,NV	I,SI	LB	V,NV	I,SI	LB	V,NV
F	--	x	x	--	--	--	--	--	x
R	--	x	x	--	--	--	--	--	x
B	x	x	x	--	--	--	--	--	x
U	x	x	x	--	--	--	--	--	x
W	--	x	x	--	--	--	--	--	x
L	--	--	--	x	--	x	--	--	--

x = Valid combination
 -- = Invalid combination

Logical Record Format

Besides the mass storage record types described earlier in this section, the following two additional record types are available for tape files.

- CYBER Record Manager control word (L)
- System block (B)

The L and B record formats are described with the other logical record formats earlier in this chapter.

Tape Formats

The tape format determines the PRU size and the definition of an LRU.

VSOS supports the following five tape formats:

- NOS internal (I)
- SCOPE internal (SI)
- Large block (LB)
- Variable (V)
- Non-ANSI Variable (NV)

I Tape Format - The NOS internal (I) tape format is the default tape format for CYBER 170 NOS systems. It is recommended that I tape format, I blocking, and L record type be used for tape interchange with NOS systems.

Within the I format, the actual PRU size can range from 6 to 3840 bytes. Each PRU is terminated by a 48-bit terminator. A 6-byte PRU is a PRU consisting solely of the terminator (a zero-length PRU). A 3840-byte PRU is a full-length PRU.

The tape hardware driver appends the PRU terminator when it writes a PRU and strips the terminator when it reads a PRU. The PRU terminator is never copied to an I/O buffer.

The end of an LRU is marked by a short PRU (less the 3840 bytes). As shown in figure 2-5, the terminator on a short PRU contains the level number of the LRU.

byte count	PRU number	level
<u>Field</u>	<u>Bits</u>	<u>Content</u>
byte count	0-11	Number of bytes in the PRU, including the PRU terminator.
PRU number	12-35	Number of PRUs since the beginning of the file.
level	36-47	Level number (0 indicates the end of an LRU, F indicates the end-of-file).

Figure 2-5. I Format PRU Terminator

A PRU consisting of only a PRU terminator containing a level number of F (17 octal) is an end-of-group indicator. The system ensures that an end-of-LRU always precedes an end-of-group indicator by writing, if necessary, a PRU terminator with a level number of zero before the end of group.

An I format tape file always contains an even multiple of bytes.

When reading an I format file, the system checks to be sure that the actual number of bytes read and the actual current PRU number match the byte count and PRU number in the PRU terminator. A mismatch is processed as a parity error.

SI Tape Format - The SCOPE internal (SI) tape format is the default format for CYBER 170 NOS/BE systems. It is recommended that SI tape format, I blocking, and L record type be used for tape interchange with NOS/BE systems.

As in the I format, the actual PRU size in the SI format can range from 6 to 3840 bytes. However, only the short PRU that ends an LRU has a PRU terminator. As shown in figure 2-6, the PRU terminator contains the level number of the LRU. If the level number is F (17 octal), the system returns end-of-group status.

The tape hardware driver appends the PRU terminator when it writes a short PRU and strips the terminator when it reads a short PRU. The PRU terminator is never copied to an I/O buffer.

The system may write 4 extra zero bits to tape in order to preserve the lower 4 bits of an 8-bit byte.

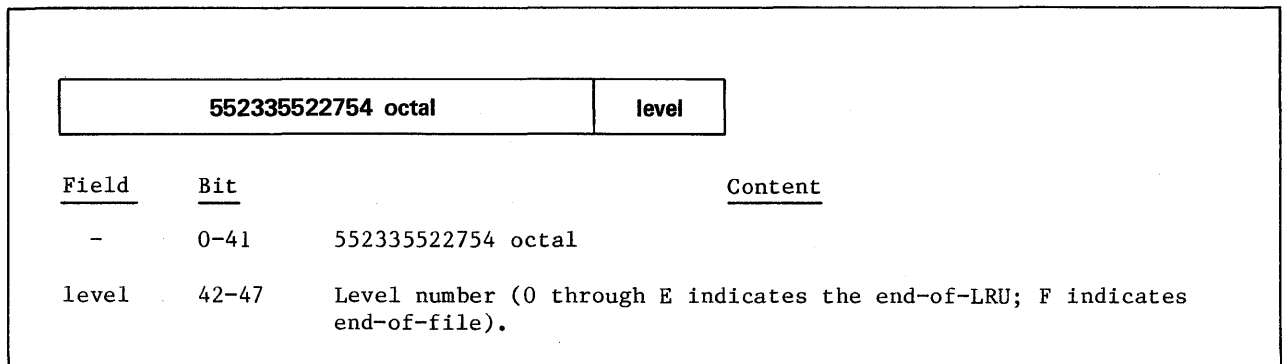


Figure 2-6. SI Format PRU Terminator

LB Tape Format - The standard PRU size for the LB format is 32768 bytes.

Each PRU in the LB tape format has a 48-bit terminator. The terminator format is shown in figure 2-7. A PRU shorter than 32768 bytes indicates the end of an LRU. If the level number is F (17 octal), the system returns end-of-group status.

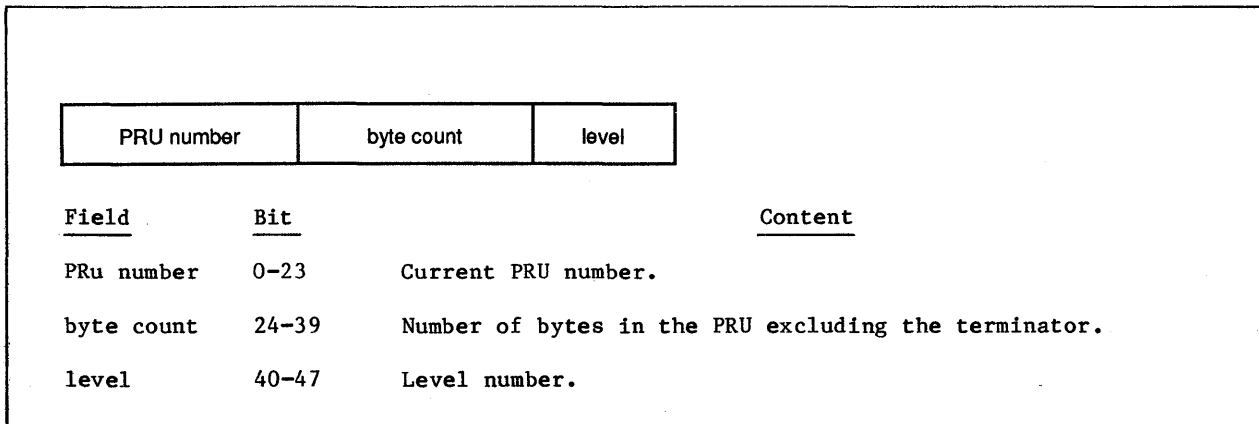


Figure 2-7. LB Format PRU Terminator

The tape hardware driver appends the PRU terminator when it writes a short PRU and strips the terminator when it reads a short PRU. The PRU terminator is never copied to an I/O buffer.

The system may write 4 extra zero bits to tape in order to preserve the lower 4 bits of an 8-bit byte.

Only C blocking is valid with the LB tape format. The L record type is not valid with the LB tape format.

V Tape Format - The V tape format can have variable PRU lengths up to the maximum PRU (MPRU) size. You may specify the MPRU size, although it cannot be greater than the buffer size (a maximum of 48 large pages). The default MPRU size is 32768 bytes (4096 words).

Within the V format, each PRU is an LRU. A PRU does not have a terminator; it has no level number associated with it.

For unlabeled V format tapes, a tape mark embedded in the file data is an end-of-group indicator if record type B is used.

K and C blocking are valid with the V format. The L record type is not valid with the V tape format.

NV Tape Format - The NV tape format is the same as the V tape format, with one exception. The NV tape format uses embedded tapemarks as end-of-group indicators for both labeled and unlabeled tapes. This allows you to write and read a labeled tape whose data does not conform to the ANSI standard.

Like the V format, the NV format can have variable PRU lengths up to the MPRU size. Each PRU is an LRU. All record types and blocking types valid for the V format are also valid for the NV format.

In addition, specifying the NEOI option on unlabelled NV tapes tells VSOS not to recognize two consecutive tape marks as EOI. This allows reading or writing tapes which do not conform to the VSOS EOI conventions for unlabelled NV tapes.

NOTE

Tasks (such as COPY and COPYL) reading tapes with the NEOI option turned on may return unpredictable errors due to reading past the end of recorded data, since no EOF or EOI status is returned by the system.

Tape Error Processing

By default, VSOS attempts to recover each error detected when reading or writing tape data. It uses the information stored in the system tapes table to reattempt the read or write operation. If it succeeds in recovering the error, the read or write operation continues.

When it encounters the end of a volume or when a file is closed, VSOS sends messages to the job dayfile and the system dayfile to report the accumulated recovered errors.

VSOS sends each of the following dayfile messages if the count in the message is not zero:

```
WRITE RECOVERABLE ERRORS=nnnn  
READ RECOVERABLE ERRORS=nnnn  
SINGLE TRACK CORRECTABLE ERRORS=nnnn  
DOUBLE TRACK CORRECTABLE ERRORS=nnnn  
DEVxxx BLOCK COUNT=nnnn
```

A listing of the system error file may be obtained from site personnel.

While reading a file, VSOS maintains a count of the blocks read. If the block count in an EOVI or EOFI label does not match the block count VSOS has maintained, VSOS sends the following messages to the job dayfile and the system dayfile:

```
DEVxxx BLOCK COUNT MISMATCH
CURRENT BLOCK COUNT=nnnn
LABEL BLOCK COUNT=nnnn
```

User Error Processing

When you request a tape file, disable standard error recovery. Disable hardware error correction for single-track errors on GCR tapes (6250 cpi).

To replace standard error recovery, specify user error processing (UEP) when opening the file with a Q5OPEN call. When you select user error processing, a tape I/O request that detects an error returns control to you, with a status code of 1476. You then call Q5GETFIT with the IOER= parameter to get the specific tape error code from the FIT. The possible tape error codes are listed in table B-4.

Now determine further processing of the error. If reading data, the program could skip forward past the record that returned the error, clear the tape error status by calling the Q5CLIOER subroutine, and continue reading the file.

CONNECTED INTERACTIVE TERMINAL FILES

A file connected to a terminal allows communication between an executing program and an interactive user. Only a task initiated by an interactive execute line can request or open a file connected to a terminal. A connected file cannot be requested or opened by a batch task or by an interactive task started by another interactive task.

Files connected to terminals are regarded as special purpose files and their characteristics are more restricted than those of other device types. These files can be used only as data files. They are always local, private files. They are sequential, with record type R, block type C, and no blank compression.

Specify the access permissions (read, write, or read and write) and security level for the connected file when requesting the file.

A program can read and write to a file connected to a terminal, using SIL Q5GETN and Q5PUTN calls. It cannot perform implicit I/O, FORTRAN unformatted or buffered I/O, or Q7BUFIN and Q7BUFOUT calls on the file. It also cannot read or write partial records to the file.

As an interactive user, you can enter record, group, and file delimiters through a file connected to a terminal. To do so, enter one of the following input lines followed by a carriage return. The \$ in the following input lines represents the system special character that prefixes request lines.

<u>Input Line</u>	<u>Program Receives</u>
\$EOR	Two consecutive record delimiters (an empty record)
\$EOG	Group delimiter
\$EOF	File delimiter

Similarly, the following output lines appear when the program outputs logical structure indicators.

<u>Output Line</u>	<u>Program Sent</u>
\$EOR	Two consecutive record delimiters (an empty record)
\$EOG	Group delimiter
\$EOF	File delimiter

The maximum length of an input or output line passed to or from the remote system is 999 characters. If a line exceeds 999 characters, it is truncated to that length, but no error condition is returned.

The following is the general processing sequence for connected files.

1. While logged in at an interactive terminal, enter a REQUEST execute line to create a file connected to the terminal. The file name must be that of the file referenced within the program.
2. Enter an execute line for the program that communicates through the connected file. The program opens the connected file.

The program can now read or write logical records to the connected file. A record written to the file appears as a line of output at the terminal. A request to read a record from the file results in a prompt (..) at the terminal.

3. To respond to a prompt, enter a line of input followed by a carriage return. The system passes the line, without blank compression, to the working storage area named on the read request.
4. The program closes the connected file when the program completes its use of the file.
5. Explicitly return the connected file, or let the system return the file when you log out.

The remote RHF application that supports interactive communication with VSOS adds a carriage return/line feed to output lines. For more information, refer to the RHF documentation for the remote system.

A task is the execution of a controllee file for a user number. A batch job or interactive session is a sequence of tasks.

INITIATING CONTROLLEE EXECUTION

An interactive or batch execute line initiates a task by naming the controllee file to be executed. The interactive and batch execute line formats are described later in this chapter.

VSOS searches for the controllee file among the files assigned to the job or interactive session. The job or session must have execute access permission to the file. To execute the file and receive dump information if its execution aborts, the job or session must have read and execute access permissions to the file.

NOTE

At security-sensitive sites, users running under a production user number can only execute production controllee files. Refer to chapter 7 of the Installation Handbook for details.

Assuming that the job or session can access the controllee file, VSOS creates the task drop file (refer to Drop Files in chapter 2) and initiates controllee execution.

VIRTUAL SPACE MAPPING

Code and data references during controllee execution are by virtual address.

The first block of the controllee file (its minus page) is not mapped to task virtual space. VSOS reads the minus page into a system table, where its information is used to control task execution. The second block of the controllee file (containing its register file) is also read into a system table.

The third and subsequent blocks of the controllee file are mapped, beginning at the origin address recorded by the LOAD utility.

LOAD also records the virtual address ranges to be used for the following areas, which are required for controllee execution but not stored in the controllee file.

- Labeled common blocks for which no space is assigned within the controllee file (blocks specified by the GROS and GROL LOAD parameters)
- Blank common

The virtual range for the dynamic stack (used for subroutine linkages and intermediate vector results) is assigned as required during task execution.

The virtual address ranges for the contents of the controllee file are initially mapped to physical space in the controllee file. However, after a controllee file page is modified, the page is mapped to the drop file, and all subsequent paging of the modified page is to the drop file. In this way, the original controllee file is never modified. The mapping of task virtual space is shown in figure 3-1.

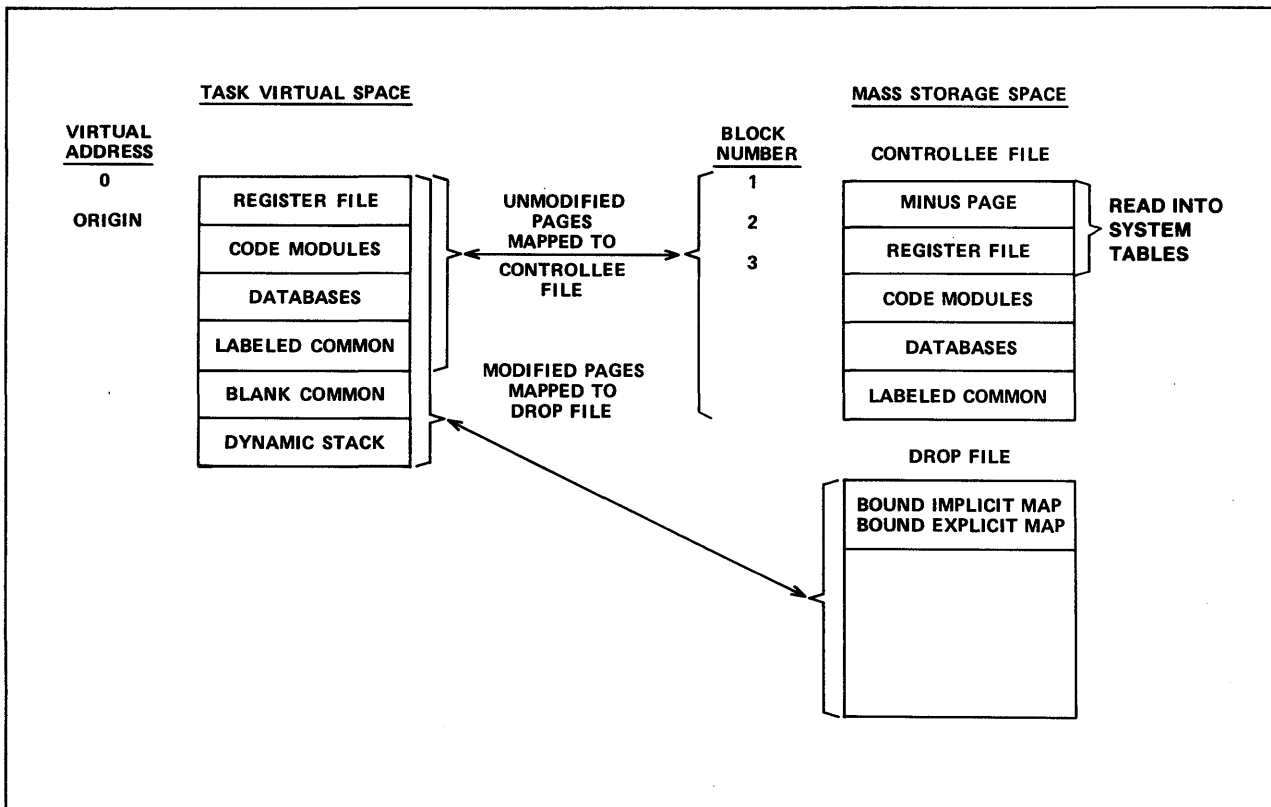


Figure 3-1. Task Mapping

CONTROLLEE CHAINS

A task can start another task. It is then the controller of the started task. The started task is the controllee of its controller.

A controller and its controllee form a controllee chain. Each task within a chain has a level. The first task (the batch processor) is at level 1; it is a controller, but not a controllee. The second task (a utility or user system) is at level 2; it is a controllee and can also be a controller if it starts a task; the task it starts is at level 3.

Except for the first and last tasks in the chain, each task in the chain is both a controller and a controllee. The lowest possible level in a chain is level 9; the maximum number of tasks in a controllee chain is nine.

In any controllee chain, only one task is eligible for execution at a time. When a controller starts a controllee, it is suspended until control is returned by the controllee.

The level 1 task of a batch controllee chain is always the batch processor. An interactive controllee chain does not have a level 1 task, because the interactive processor in the virtual system is not a task. However, for consistency, the Q5LSTCH routine described in chapter 8 of this manual lists the interactive processor as the level 1 task of an interactive controllee chain.

As described later in this chapter, a job is a sequence of level 2 tasks initiated by the batch processor.

SYSTEM ACCESS

To start a task, access VSOS. VSOS supports both batch and interactive access.

To access VSOS interactively, log in at an interactive terminal connected to a front-end system. After logging in to the front-end system, log in to VSOS.

For batch access to VSOS, send a VSOS batch job file from a front-end system to the CYBER 200 system.

USER VALIDATION

Both batch access and interactive access require user validation. User validation is performed when VSOS processes a LOGIN line or a USER statement.

User validation requires entry of a valid user number. Entry of the password for the user number is also required unless the password is blank.

The user number determines your privileges and the files that you can access. The account identifier can be used to charge you for resource usage. The account identifier default is the account number to which the operating system automatically charges the usage.

The VSOS user directory defines the valid VSOS user numbers. Each valid user number has an entry containing the number, its password, and the validations for the user number. The following are the user number attributes.

- Permission to execute high priority tasks
- Permission to perform privileged functions
- Permission to use the variable rate accounting rate feature (for more information, refer to volume 2 of this manual)
- Permission to access tapes via batch processing and/or interactive sessions
- Permission to access the system interactively
- Maximum security level (refer to File Security Levels in chapter 2 of this manual)
- Valid job categories (refer to Resource Allocation in this chapter)
- Valid account identifiers (default account number, master and/or optional account numbers)
- System seconds available (refer to Accounting in this chapter)
- Default project number
- User project control

Only the installation management user number can change the user directory. For user number information, ask site personnel.

INTERACTIVE SYSTEM ACCESS

VSOS supports interactive access through RHF. In each case, perform the following steps:

1. Log in to the front-end system.
2. Request that the front-end RHF software provide a connection between the interactive terminal and the CYBER 200 system.
3. Log in to VSOS.

The statements required to request a connection to the CYBER 200 system differ, depending on the front-end system software. The RHF interactive access statements are described in the RHF manual for the front-end system.

VSOS INTERACTIVE LOGIN

After connecting to the CYBER 200 system, log in to VSOS with a LOGIN command.

The LOGIN command identifies you. The system checks to be sure that the specified user number is a valid user number and that the specified password is the correct password for your user number.

The LOGIN command connects you to a job descriptor number (JDN).

The LOGIN command may specify the account identifier to which resource usage during the session is charged. It may also specify the security level of the session. A task started during the session cannot access a file with a security level greater than the session security level.

If you logged out or were disconnected while a task was still active, the LOGIN reconnects you to the existing job descriptor number. If no tasks were running when you logged out or were disconnected, a subsequent LOGIN connects you to a new job descriptor number.

The format of the LOGIN command is shown in figure 3-2. A blank, comma, or left parenthesis must follow the LOGIN verb. A terminating period or right parenthesis is valid but not required. To send the LOGIN line, press carriage return.

The LOGIN parameters must appear in the order shown in figure 3-2. The parameter separator is either a blank or a comma. Except for the last parameter, parameter omission is indicated by two consecutive commas.

LOGIN, userno, password, account, n

userno	User number (six decimal digits). This parameter is required.
password	Password (one to eight characters). To specify a blank password, omit the parameter. To enter a password that contains special characters, refer to the discussion of the PASSWORD control statement in chapter 4 of this manual. Site personnel determine during system installation whether user password entry is required or optional.
account	Account identifier (one to eight characters). This parameter is optional. If an account identifier is omitted, the user's designated default account identifier is used.
n	Optional security level of the interactive session (1 to 8). If a security level is omitted, the default value chosen by the site is used.

Figure 3-2. LOGIN Command Format

BATCH SYSTEM ACCESS

VSOS processes input in batch mode when it is received as a batch input file.

The CYBER 200 RHF application program, QTFS, receives batch input files sent by RHF software on front-end systems. QTFS accepts multiple files on one connection. A front-end system that uses RHF software is called a remote system.

QTFS performs the following functions:

1. Copies the batch input file as R format records to a CYBER 200 mass storage file
2. Validates the user submitting the batch job
3. Gives the batch input file to the input queue manager

QTFS creates a mass storage file that has an eight-character name. The name is derived by padding to eight characters the job name supplied by the remote system and ensuring that the name is unique in the input queue and different from the input job name. To get a unique name, increment the rightmost characters (A to B to C to . . . Z).

After storing the batch input file in the mass storage file it creates, QTFS reads the USER statement from the file. The USER statement must be the second statement in the file. The first statement is used by the remote system (refer to RHF documentation for the remote system).

QTFS uses the USER statement information to validate your access to the CYBER 200 system. Assuming that you are a valid user, QTFS removes the first two statements from the file before it gives the file to the input queue manager (IQM).

IQM processes the RESOURCE statement if you included one in the job. The RESOURCE statement must be the third statement in the batch input file if one is used. RESOURCE statement processing is described under Resource Allocation in this chapter.

All other control statements for the job follow the RESOURCE statement. Batch job structure and the processing of a batch job are described later in this chapter.

INTERACTIVE SESSION

VSOS login initiates an interactive terminal session. Enter a \$BYE request line to end the session.

To enter an interactive input line, type the line at the terminal keyboard and then press the carriage return key to send the line to VSOS.

Enter the following types of interactive entries:

- Break character
- Interactive terminal request
- Interactive execute line
- Input to an executing task

BREAK CHARACTER

The break character terminates the task. It transfers control to the immediate controller of the task.

To terminate a task within a batch job, you must use the \$SU command to determine the JDN and job name of the job, then request that the operator drop or kill the job. Control returns to the batch processor, which searches for an EXIT statement in the control statement sequence of the job. If it finds one, it continues execution after the EXIT statement.

If the terminated task is an interactive task with no controller, it can be restarted by executing its drop file (refer to Drop Files in chapter 2).

The default break character is !, but it can be changed during system installation.

INTERACTIVE REQUEST LINES

An interactive request line is a special entry that the interactive processor executes without initiating a new task.

The interactive processor recognizes a request line when an input line begins with the special character. By default, the special character is \$, although the site can change the special character during system installation. You may change the special character during a terminal session; the procedure to follow is described next.

Changing the Interactive Request Special Character

The following request line changes the interactive request line special character:

```
$=sc
```

The single character represented by sc becomes the new special character that must begin all interactive request lines. VSOS responds with:

(SC)=sc

where sc is the new special character. The new character must be a printable ASCII character between #21 (!) and #7E (~). If the new character is not in this range, the special character is not changed and VSOS responds with the message:

BAD (SC)

Terminal Information Requests

The following request lines list information at the terminal.

<u>Request Line</u>	<u>Information Returned</u>
\$T	Current date and time in format mm/dd hh.mm.ss; mm/dd gives the month and day and hh.mm.ss gives the hour, minute, and second.
\$S	Current state of the program executing under the current JDN of the interactive session. Table 3-1 lists the possible responses.
\$BB	Current accounting information for the program executing under the current JDN of the interactive session. It lists the remaining time available.
\$?	Current date, time, accounting information, program state, and job descriptor number of the executing program.
\$SU	Current activity, job descriptor number, and job/session name for all of the user's jobs in execution.
\$PR	Number of interactive tasks waiting for execution.
\$P	Pools attached to the user's interactive session.

Table 3-1. Program States (Sheet 1 of 2)

Response	Meaning
RUNNING	In execution.
WAIT ALT	Waiting for CPU assignment.
WAIT TPE	Waiting for tape assignment.
WRT CNTR	Waiting for controller to be assigned initial memory resources.
WRT CNTE	Waiting for controllee to be assigned initial memory resources.
RCV CNTR	Waiting for message from controller.

Table 3-1. Program States (Sheet 2 of 2)

Response	Meaning
RCV CNTE	Waiting for message from controllee.
SND CNTR	Waiting to send message to controller.
SND CNTE	Waiting to send message to controllee.
SND OPR	Waiting to send message to operator.
SND TTY	Waiting to send message to teletype.
DUMPING	Input/output being dumped to disk.
FINISH	Finished; clean-up is in progress.
SUSP OPER	Suspended by system operator.
SUSP SYS	Suspended by system.
WAIT MP	Waiting for minus page to be assigned.
RCV OPER	Waiting to receive message from operator.
WAIT mfx	Waiting for mainframe identified.
NIL	No tasks in execution.

Case Conversion Request

When a terminal session starts, VSOS translates all input to uppercase before processing. The following three request lines control this action.

<u>Request Line</u>	<u>Information Returned</u>
\$LC	Enable lowercase input. VSOS will not translate characters to uppercase.
\$UC	Disable lowercase input. VSOS will translate characters to uppercase.
\$X	Toggle case translation. If VSOS was translating characters to uppercase, it will no longer do so. If VSOS was not translating characters to uppercase, it will begin to.

The response to these requests is either a lowercase L or an uppercase U. If the lowercase L is returned, VSOS will not be translating characters to uppercase. If the uppercase U is returned, VSOS will be translating characters to uppercase.

VSOS understands all interactive request lines and connected terminal file delimiters whether case conversion is being performed or not.

Operator Message Request

The following request line sends a message to the system operator in the K display:

\$OP message

When the message is sent, the system responds with the following line:

[OK]

Task Interrupt Request

The following request line can interrupt and send a message to the interactive task if the task has been designed to receive this input:

\$I message

The message is optional. A \$I entry interrupts the task without sending a message.

To be able to process the message interrupt, the program must include the following:

- A call to enable interactive message interrupt processing (a Q5ENAMI call with the TERMINAL parameter specified).
- The message interrupt subroutine specified on the Q5ENAMI call. The subroutine can contain one or more Q5GETMCR calls to receive \$I input messages and one or more Q5SNDMCR calls to send messages to the terminal. (The interrupt routine usually reads and/or sets variables in a common block to monitor or alter the execution of the main program.) It must contain a Q5RFI call to end interrupt processing.

After you start a task that uses a message interrupt subroutine, a message can be sent at any time to the interrupt subroutine using a \$I interactive request line.

A \$I request line interrupts the task. The current state of the interrupted task is saved, and the interrupt routine specified on the Q5ENAMI call is entered. The interrupt routine can get the \$I message by calling Q5GETMCR. Control returns to the task from the interrupt routine when it calls Q5RFI; the task resumes its processing as if no interrupt occurred.

The Q5ENAMI description in chapter 8 contains an example of a message interrupt routine.

Session Termination Request

The following request line terminates the VSOS interactive session.

\$BYE

The \$BYE request line does not terminate active tasks belonging to the user number; the tasks execute until they are completed. In a system with an RHF front-end, the \$BYE request line causes control of the terminal to be returned to the front-end system.

The \$BYE request line indicates that after all active tasks are completed, the system can discard the local files created during the interactive session. If the session is disconnected without a \$BYE request line, the local files are not discarded. Logging in with the same user number reconnects you to the existing JDN.

When interactive access to VSOS is via RHF, it is possible to log out from VSOS without returning control of the terminal to the front-end system. This is done by typing the following command:

\$HELLO

The \$HELLO request line has the same effect as \$BYE, except that control of the terminal is not returned to the front-end system. Instead, a prompt to log in is issued, and a LOGIN command may be entered. This feature is useful for switching between two or more numbers with a minimum of effort.

INTERACTIVE EXECUTE LINE

Start execution of a controllee file by logging in to an interactive terminal. Enter an execute line with the general format shown in figure 3-3.

The execute line must specify the name of the file to be executed. The file can be a local file, an attached private file, or a public file. The file name becomes the task name.

The execute line can also specify resource limits for the task and a character string to be passed to the task. The character string can be entered before or after the resource parameters. If resource parameters are specified, a (blank)/(blank) must precede the parameters and a second (blank)/(blank) must separate the parameters from the string if it follows the parameters.

All parameters specified should conform to the conventions used on system-supplied control statements. All addresses are assumed to be hexadecimal values; any other number is assumed to be a decimal value unless preceded by #.

NOTE

Many VSOS utilities are case sensitive for input or command lines. If case conversion is not being done by VSOS, you will have to enter data in uppercase where required.

taskname / TL=t,PRIORITY=p,WS=w,LP=lp / string

or

taskname string / TL=t,PRIORITY=p,WS=w,LP=lp

taskname Name of task to be placed into execution (one through eight letters or digits).

string Optional character string to be passed to the task. The format of the character string depends on the coding of the task. The string is delimited by blanks. The delimiting blanks are not passed to the task.

Optional Resource Parameters

TL=t Task time limit in system seconds† (decimal integer between 1 and 599940).

If TL=t is omitted, the task time limit is determined by an installation parameter value (release value, 60).

PRIORITY=p Task priority, 1 (lowest) to 15 (highest). If the specified priority exceeds the maximum priority specified by the installation for interactive tasks, the task priority is set equal to the maximum priority.

If PRIORITY=p is omitted, the job priority is the installation-specified default priority for interactive tasks (release value, 14).

WS=w Maximum working set size in blocks (decimal integer).

Specifying WS=* notifies the system that the task requires all allocatable memory (a machine size task). If the site does not allow an interactive task to use all allocatable memory, the task is not started.

If WS=w is omitted, the maximum working set size for the task is the maximum working set size for interactive tasks.

†A system second is one million STUs. If desired, an installation can substitute SBUs for system seconds as the time limit unit. The calculation of an STU or an SBU is described in volume 2 of this manual.

Figure 3-3. Interactive Execute Line Format (Sheet 1 of 2)

NOTE

Use the WS parameter only for the following tasks:

- A machine size task requiring all allocatable memory (specify WS=*)
- A task known to execute efficiently with a maximum working set size for interactive tasks

Misuse of the parameter could result in suspension of the task to prevent system performance degradation. The system automatically resumes the task when system resources are available.

LP=lp

Maximum number of large pages that can be assigned to the task (decimal integer). If the specified limit exceeds the maximum limit specified by the installation for interactive tasks, the task is not started.

If the large page limit, when multiplied by 128, exceeds the working set size limit, the task is aborted.

If LP=lp is omitted, the large page limit is zero.

Figure 3-3. Interactive Execute Line Format (Sheet 2 of 2)

TASK DATA INPUT

Tasks can be programmed to accept input from a file connected to a terminal. Use of a connected file is described in chapter 2 of this manual under Connected Interactive Terminal Files.

DYNAMIC AND STATIC EXECUTION

A controllee may execute statically or dynamically. A controllee executed statically has all externals loaded at load time. The controllee can be loaded for dynamic execution. Dynamic execution makes use of a linker utility that satisfies externals on a dynamic basis when the controllee is executed. The linker utility loads dynamic modules and utilities from a user dynamic or the system shared library (SHRLIB). Dynamic utilities are system utilities that do not have any SYSLIB modules on their controllee files. These SYSLIB modules are called dynamically during execution. All system utilities are dynamic except the following:

DEBUG
CTX
META

The SHRLIB allows both batch and interactive users to share the same physical pages in virtual memory. The system sets aside enough physical pages to satisfy the working set of SHRLIB. The SHRLIB is a file that is read or partially read during system initialization into the pages set aside for the shared library region of memory. Those pages are unavailable for other use.

The shared pages contain directories, a dynamic linker, and a shared SYSLIB that contains dynamic modules and, optionally, shared utilities.

The only shared utilities are BATCHPRO and FTN200. During execution of a controllee, if a controllee faults for a page of the SHRLIB that is not in memory, the system reads in the page for the controllee to use.

For more information on dynamic execution, refer to the LOAD utility in chapter 4 of this manual.

BATCH JOB

A VSOS batch job is a sequence of tasks the batch processor starts while it executes for a user number. The sequence of tasks is specified in a batch input file.

BATCH INPUT FILE STRUCTURE

A batch input file comprises one or more groups of records. The first group contains a sequence of batch execute lines (control statements) specifying the sequence of tasks to be performed. Each execute line is a separate record.

Subsequent groups in the file can contain input for tasks initiated by execute lines in the first group. Input could be source program text, program data, or utility directives.

The input groups must be in the order required by the execute line sequence. For example, suppose the sequence includes the following statements:

```
FTN200.  
LOAD.  
GO.
```

The input groups must include a source program text group as input for the FTN200 task, followed by a group containing any data required for the GO task. The LOAD task does not require input from a batch input file group.

Only one execute line can use a group from the batch input file. For example, a source program text in the batch input file can be used by only one compilation statement in the job.

If a RESOURCE statement is included in the job, it must be the first statement after the USER statement, if specified. SUBMIT or QTFS processes the RESOURCE statement. The USER statement is always the first control statement after the job statement. The job statement is the first statement in a VSOS batch job. The job statement must begin with a 1 through 8 alphanumeric character string with the first letter being alphabetic. The job statement must end with a valid job control statement terminator. SUBMIT/QTFS removes the job, user, and RESOURCE (if present) statements from the job file prior to placing it in the input queue.

BATCH CONTROL STATEMENT

A batch control statement initiates a task within a batch job.

The following is the general format of a batch control statement.

```
task-name,parameters. comment
```

Any blanks before the task name are ignored. The task name can be followed by any of the following separator characters:

```
( , blank
```

If the task name and a parameter are separated by more than one blank, only one blank is passed to the task.

If a parameter (such as the JCS parameter in the MFLINK statement) specifies a character string, double quotes (") must delimit the string. To include the character " in the string, two consecutive quotes must be specified. The two " characters are interpreted as one " character within the string.

A control statement is normally translated to uppercase characters before being processed. However, if you put character strings within double quotes, this translation is not done. Thus, you have a way to supply case-sensitive arguments to a task.

A control statement must be terminated by either a right parenthesis or a period. A right parenthesis or period specified within a character string does not terminate a control statement. Blanks to the right of the terminator are ignored.

If a terminator does not appear in the line, the line immediately following it is presumed to be a continuation. (A COMMENT or * control statement cannot be continued.) No special continuation character exists for a batch control statement.

Any characters after the terminator are presumed to be a comment. These characters are copied to the dayfile but are not otherwise processed.

The parameters of a batch control statement must be checked by the task; the batch processor does not interpret the parameter string.

The batch input file should not contain an execute line that returns it. If the input file is returned, it is not purged at the end of the job. This causes the job to be resubmitted to the input queue at each autoloading.

JOB SCHEDULING

Each batch input file entered in the system is processed by the input queue manager. It assigns each batch job a job selection number that determines its position in the input queue. The job selection number is based on the job priority and on the time the job entered the system. The batch user can specify a priority on the RESOURCE statement (refer to chapter 4 in this manual). Jobs with the same priority are positioned in the queue according to the time they entered the system; older jobs have a higher job selection number.

When an executing job or task has terminated, the input queue manager determines the next job to give to the CPU scheduler. Starting with the job with the lowest job selection number, the input queue manager selects the first job in the queue that meets scheduling constraints. The Q control statement (refer to chapter 4 of this manual) lists the status of the jobs in the system. A job may wait in the input queue for the following reasons:

- The maximum number of executing jobs for the job category has been reached.
- There are not enough uncommitted tape drives available to satisfy the tape drive requirement for the job.
- Reservation of the requested maximum working set size would overcommit memory beyond the allowed overcommitment percentage.
- Adding the job's time limit to the sum of the time limits of all executing jobs would exceed the maximum time limit for all jobs.

These conditions are self-correcting; that is, eventually the condition preventing job execution ends, and the job leaves the input queue. However, other conditions that can hold a job in the input queue are not self-correcting. In the following four cases, the job remains in the input queue until the operator enters a command to remove the job.

- The job category for the job is turned off.
- The operator has entered a command to hold the job in the input queue.

- The number of tapes requested for the job is greater than the number of tape drives available at any time on the system.
- The job requires tape files, but the operator has turned off tape processing.

In the following three cases, the job does not enter the input queue. The job returns to the remote host and the dayfile contains the following message:

NO JOB CATEGORY FOUND FOR SPECIFIED LIMITS

Operator commands are ineffective in trying to change the job category limits to prevent the job from aborting.

- The job's maximum working set size is greater than the maximum working set size limit for its job category.
- The job's large page limit is greater than the maximum large page limit for its job category.
- The job's time limit is greater than the maximum job time limit for its job category.

Interactive tasks go directly to the CPU scheduler without processing by the input queue manager.

JOB PROCESSING

The batch processor processes a job while executing under the user number of the job. It uses the batch input file as its input. The batch input file is a permanent file under the submitter's user number. Its name is derived by padding to eight characters the job name supplied on the front-end job card, ensuring that the name is unique within the input queue and different from the input job name. To process a job, the batch processor performs the following steps:

1. It creates a file named Q5JOBFLE and copies the first group in the batch input file (the control statements) to it.
2. It creates a file named Q5JRTHRF and copies the last group in the batch input file to it.
3. It creates a file named INPUT and copies the next group in the batch input file (if one exists) to it.
4. It creates a file named Q5DAYFLE for the job dayfile (refer to Job Dayfile in this chapter for more information).
5. It sets the error threshold value at the default value of 4. (The threshold value can be changed by a TV control statement.)
6. It reads a record from Q5JOBFLE. If it reads the end of the file, it initiates normal job termination. The batch input file is purged upon completion of the job.
7. It determines whether the statement in the record is for a batch processor function. If it is, it executes the function and continues job processing at step 5.

8. If the statement is not for a batch processor function, it searches for the file having the name specified on the execute line. (Refer to File Search Hierarchy in chapter 2 of this manual.)

If the search fails to find an executable file with the specified name, the batch processor aborts the job.

9. If the search finds an executable file with the specified name, the batch processor starts the task as its controllee, passing to it the parameters on the statement.
10. When the task is completed, the batch processor receives the completion status of the task. If the task abort flag is set, the batch processor aborts the job.
11. If the task did not set its abort flag, the batch processor compares the return code returned by the task (its termination value) with the error threshold value for the job. If the termination value is greater than the threshold value, the batch processor initiates abnormal job termination.
12. If the termination value is not greater than the threshold value and the task generated an OUTPUT file, the batch processor renames the OUTPUT file as a member of the print file family for the job. It is named Pnnfammm. nn is a sequence number starting with 00, and fammm is the unique family identifier.
13. Returns the current INPUT file if the task terminated normally and the INPUT file was read by the task. It then creates a new INPUT file and copies the next group in the batch input file (if one exists) to the new INPUT file.
14. Job processing continues at step 5.

JOB DAYFILE

For each job the batch processor processes, a job dayfile is created. During job processing, the batch processor, the operator, and the executing tasks record job events, job status, and comment and error information in the job dayfile.

The batch processor records all operator commands that relate to the job, messages the task sends to the batch processor, and messages the job or the operator send to the dayfile. The user can send a message to the job dayfile with a COMMENT control statement or a SIL Q5SNDMDF call.

The job dayfile is printed at the end of the job output.

JOB TERMINATION

A job terminates when one of the following events occurs.

<u>Event</u>	<u>Action</u>
A task sets its abort flag.	Job abort is processed, followed by abnormal termination processing and the job termination procedure.
A task returns a termination value greater than the job threshold value.	Abnormal job termination is processed, followed by the job termination procedure.

<u>Event</u>	<u>Action</u>
The end of the Q5JOBFILE file is read.	Job termination procedure is processed.
Exit control statement is read.	Job termination procedure is processed (refer to the EXIT control statement description in chapter 4 of this manual).

Job Termination Procedure

The batch processor always performs the following steps to terminate a job, regardless of why the job terminates.

1. It renames the Q5DAYFILE file so that it is a member of the print file family of the job. It is named PXXfammm.

If the input queue manager received the job from CYBER 200 RHF, a final print file named PYYfammm is added to the print file family. It contains the RHF routing information for the file.

2. It processes the print file family of the job for routing to a front-end system, as described under Print Files in chapter 2 of this manual.
3. It destroys the batch input file.

Job Abort

If the task abort flag is set, the batch processor performs the following steps:

1. It dumps task information (refer to DUMP in chapter 6 of this manual). To receive a dump, the file being executed must have read access permission.
2. It initiates abnormal job termination.

Abnormal Job Termination

When abnormal job termination is initiated, the batch processor searches for the next EXIT or PROCEED control statement in the job file. If it finds an EXIT or PROCEED statement, it continues processing with the statement following that statement.

If the batch processor does not find an EXIT or PROCEED statement, it routes the job output and terminates the job.

JOB PROCESSING EXAMPLE

Suppose the sequence of information shown in figure 3-4 is an input file for the batch processor. (The input queue manager has removed and processed the RESOURCE statement.)

```

TV,0+.
FTN200.
LOAD.
GO.
DEFINE,DATAOUT.
end-of-group delimiter
.
.
FORTRAN 200 source program
.
.
end-of-group delimiter
.
.
program data
.
.
end-of-file delimiter

```

Figure 3-4. Example Batch Input File as Read by the Batch Processor

The batch processor begins to process the file by copying the first group of the file (containing the control statements) to a file named Q5JOBFILE. If the batch job originated from a remote system or via SUBMIT, the last group in the file is copied to a file named Q5JRTHRF. The processor then copies the next group (containing a FORTRAN source program) to a file named INPUT. It also creates a file named Q5DAYFILE and sets the error threshold value to the default value.

The batch processor reads the first record from Q5JOBFILE. It contains the batch processor control statement, TV,0+. The batch processor executes the statement, setting the job error threshold value at zero.

The batch processor then reads the second record from Q5JOBFILE. It is the compiler control statement FTN200. The batch processor does not recognize the statement as one that it executes, so it assumes that the statement names a controllee file to be executed. It passes the parameters of the control statement and the file name to the operating system.

The operating system searches for a controllee file with the name FTN200. If the system finds a controllee file with that name, the batch processor starts execution of the file as its controllee.

Assuming that the controllee is an FTN200 compiler, the compiler, by default, reads its input from the INPUT file containing the batch input file group that followed the control statement group. Assuming that the group is an FTN200 source program, the compiler, by default, creates the local file BINARY and writes the compiled object code on the file. The compiler also creates a local file named OUTPUT, on which it writes the FTN200 source listing.

Assume that the FTN200 task returns a completion status of 0 (normal termination) to the batch processor. Because the INPUT file was read by the FTN200 compiler, the batch processor returns the existing INPUT file and creates a new INPUT file, copying the next group from the batch input file.

The batch processor changes the name of file OUTPUT to P00fammm. The name fammm is the unique identifier of the family of print files belonging to the job.

The next record the batch processor reads from Q5JOBFLE contains the control statement LOAD. The operating system finds the public file LOAD, containing the LOAD utility. It initiates execution of the file as a controllee of the batch processor.

By default, LOAD reads the BINARY file as its input. It creates a local file named GO, on which it writes a controllee file. It also creates a local file named OUTPUT, on which it writes the load map.

LOAD attempts to satisfy external references from the default library SSYSLIB. Remaining unsatisfied externals are assumed to be dynamic and will be satisfied by the linker when the GO file is executed.

At the end of LOAD execution, the batch processor again checks the abort flag and compares the termination value to the error threshold value. LOAD did not read the INPUT file, so the INPUT file is not returned. LOAD did create an OUTPUT file, so the batch processor changes the name of the OUTPUT file to P01fammm.

The next control statement read from Q5JOBFLE is GO. The operating system finds the GO file as the local controllee file the LOAD created. It executes GO as a controllee of the batch processor.

At the end of GO execution, the batch processor checks the abort flag and compares the termination value to the error threshold value. GO read the INPUT file, so the batch processor returns the INPUT file, but no more input groups exist on the batch input file, so a new INPUT file is not created. GO changes the name of the OUTPUT file created by GO to P02fammm.

The next control statement read is DEFINE,DATAOUT. Execution of the DEFINE utility searches for a local file named DATAOUT. Assuming that execution of GO created a file named DATAOUT, DEFINE stores the local file DATAOUT as a permanent file. Therefore, the file DATAOUT continues to exist after job termination; all other files used by the job are destroyed at job termination.

Q5JOBFLE contains no more records to be read, so the batch processor terminates the job.

The job dayfile, Q5DAYFLE, created as the job was running, is changed to PXXfammm. If it exists, the last-group-file Q5JRTHRF, is renamed PYYfammm. If the job originated from a remote system or via SUBMIT, all files in the output-file-family are given to the output queue as they are renamed.

REMOTE HOST FACILITY

The remote host facility (RHF) is the set of software that enables communications between computer systems connected via the loosely coupled network (LCN) hardware. RHF manages the transfer of control statements and files between systems. It also supports VSOS interactive access from remote systems. It performs all required character code and logical file structure conversion.

The RHF software that executes on a CYBER 200 system is called CYBER 200 RHF. The software includes an application program for each RHF function. These application programs send control statements to and receive control statements from RHF application programs executing on other computer systems.

The applications and CYBER 200 RHF control statements executed by the system are described in this chapter. For a description of the user-executable RHF applications, DUMPF, LOADPF, MFLINK and MFQUEUE, refer to chapter 4 of this manual.

RHF can send control statements to another system to be executed by the other system. The control statements are specified as a text string. The text string can be specified as part of a parameter or as data on a separate file. CYBER 200 RHF takes the control statement text string from the JCS parameter or from the file specified on the INPUT parameter. For security reasons, a text string specified on a JCS parameter is not entered in the job dayfile. The text string is replaced by asterisks.

RHF can transfer copies of queue files, permanent mass storage files, or archived files. A separate RHF application program manages each file transfer category. The following are the CYBER 200 RHF applications.

<u>Program</u>	<u>Description</u>
Interactive Transfer Facility Server (ITFS)	Manages interactive I/O transfers.
Queue File Transfer Facility (QTF, QTFS)	Manages queue file transfers.
Permanent File Transfer Facility (PTF, PTFS)	Manages requests for permanent file operations, including permanent file transfers.
Dump/Load Facility (DLF)	Manages dumping and reloading of CYBER 200 archived files.

The remote operator interface is described in the VSOS 2 Operator's Guide.

INTERACTIVE ACCESS

To log in to the CYBER 200 system via RHF, perform the following steps:

1. Log in to the remote system.
2. Notify RHF that you want to log in to the CYBER 200 system, as described in the RHF documentation for the remote system.
3. Log in to VSOS as described earlier in this chapter.

After login, interactive access via RHF is as described earlier in this chapter.

QUEUE FILE TRANSFERS

The QTF and QTFS applications manage queue file transfers for VSOS. QTFS receives batch input queue files with disposition codes of IN and IX and gives them to the input queue manager as CYBER 200 batch jobs. QTFS also receives output queue files with disposition codes of LP, CP, P8, PB, or SP, and gives them to user 6 for processing by QTF. QTFS will also process the following routing directive if received with the queue file:

```
ROUTE,ST=lid.
```

lid. The logical id of the remote host to receive any output associated with the queue file.

The remote system's MFQUEUE command is used to send the routing directive with the queue file. For example, if the ROUTE directive is received with an input queue file, the output of the batch job is sent to the LID specified in the ROUTE directive. If the ROUTE directive is received with an output queue file, the output queue file is given to user 6 and QTF sends the output queue file to the LID specified in the ROUTE directive.

If QTFS receives an output queue file without any routing directives, QTFS gives the output to user 6 and QTF sends the output to a default remote host. Also, if QTFS receives an input queue file with disposition of IX and no routing directives are received with it, the output generated by the batch job is sent to a default remote host. The default remote host is designated initially by an LID specified by the AUTOCON variable OLID (the released value is NOS; on VSOS it is ME1). However, the remote host may be changed when the system is autoloaded or by the OLID operator command during system operation.

The QTF application sends queue files to other remote hosts sending information with the queue file designating the file's disposition.

CYBER 200 JOB SUBMISSION

QTFS receives a batch input file for processing as a CYBER 200 job. The job statement is the first statement in the file and is used by the remote system. QTFS processes the second statement in the file, the USER statement. If a RESOURCE statement follows the USER statement, QTFS processes it also. QTFS then strips the first two statements from the file and the RESOURCE statement (if present) and gives the file to the VSOS input queue manager.

The USER control statement (refer to figure 3-5) identifies the CYBER 200 user number to which the CYBER 200 job belongs, the account identifier to which its resource usage is charged (optional), the password for the user number, and the security level of the job.

The RESOURCE control statement for the job must immediately follow the USER statement.

The logical structure of a job file sent to the CYBER 200 input queue must be indicated by ASCII separator characters. The job file may be transferred to the CYBER 200 only in character mode (DD=C8, C6, or the default, which causes the file to be handled in the native character mode of both the sending and receiving systems).

USER,USER=userno,ACCOUNT=account,PASSWORD=password,SECURITY=n.

<u>USER</u> =userno	CYBER 200 user number (one to six decimal digits). This parameter is required.
<u>ACCOUNT</u> =account	CYBER 200 account identifier (one to eight ASCII characters). This parameter is optional.
<u>PASSWORD</u> =password	User password (one to eight ASCII characters). Site personnel determine during system installation whether user password entry is required or optional.
<u>SECURITY</u> =n	Security level for the job (1 to 8). If SECURITY=n is omitted, security level 1 is assumed.

Figure 3-5. USER Control Statement Format

OUTPUT FILE ROUTING

Because a CYBER 200 system has no output devices, RHF must transfer all output files to a remote system. The QTF application performs all output file routing.

As described under Job Processing earlier in this chapter, the final file in an output-file-family contains the output specifications for the files. The files in the family, including the job dayfile, are concatenated at job termination into a single print file. When the RHF software on the remote system receives the file, it interprets the output specifications to determine the appropriate output queue for the file.

The records of the routed file are terminated by ASCII unit separator characters (#1F). RHF adds these record termination characters, if necessary, as part of the output file.

EXPLICIT FILE ROUTING

By default, QTF routes an output file to the remote system from which the job that produces the output file originated. However, an output file can be explicitly routed to another system. To do so, include an MFQUEUE control statement (refer to chapter 4 of this manual) in the job that produces the file.

RHF PERMANENT FILE REQUESTS

RHF permanent file requests are handled by a sequence of control statements sent to or from a remote operating system via RHF. The PTF and PTFS applications manage RHF requests for permanent file operations. PTFS receives requests from the remote system for operations on permanent files. PTF sends requests to the remote system, using the MFLINK control statement, for operations on permanent files.

PERMANENT FILE REQUESTS

PTFS receives requests to define, copy, purge, and give CYBER 200 permanent files. The following VSOS control statements are acceptable to PTFS:

ATTACH
AUDIT
CHARGE
DEFINE
GIVE
MFGIVE
MFTAKE
PATTACH
PDETACH
PERMIT
PURGE
Q (only if IP_SCF = 1)
RETURN
SWITCH
USER

The sequence of CYBER 200 control statements that compose a request must adhere to the following rules:

- It must begin with a USER statement (refer to figure 3-5) specifying the CYBER 200 user number to which the referenced files belong.
- To copy a permanent file, the sequence must include the statements needed to access the CYBER 200 file and either an MFGIVE or an MFTAKE control statement (refer to figures 3-6 and 3-7). An MFGIVE statement copies a CYBER 200 file to a remote file. An MFTAKE statement copies a remote file to the CYBER 200 file.
- Only one MFGIVE or MFTAKE control statement can be specified per request.
- You may omit the ST=lid parameter and the USER card in the JCS parameter string for second and subsequent MFLINKs in the same job or interactive session on the front end. In batch jobs, this allows multiple files on one connection for successive MFLINKs.

If you use multiple copies of PTFS referencing the same permanent file, you may experience delays on the ATTACH. The default for ATTACH is to wait until access is granted to the file. Default access permissions are RX (read and execute). Several jobs may access the file simultaneously to read it, but if one PTFS specifies write access permission on the ATTACH statement, on all subsequent attempts to ATTACH the file with any access, the job waits until the file becomes available (unless WAIT=NO was also specified). If the wait interval becomes too long, MFLINK on the remote system may time out. See chapter 4 of this manual for additional information on the ATTACH statement. If the remote system sends successive requests on a single MFLINK connection (MFLINK session), PTFS allows the requests and completes the session.

MFGIVE, lfn.	
lfn	Name of the CYBER 200 file to be transmitted (one to eight alphanumeric characters, beginning with a letter). The statements required for accessing a permanent file (ATTACH or PATTACH) must be executed before the MFGIVE statement.

Figure 3-6. MFGIVE Control Statement Format

MFTAKE, lfn.	
lfn	Name of the CYBER 200 file that receives the file copy. To make the file permanent, the DEFINE statement must be executed before or after the MFTAKE statement.

Figure 3-7. MFTAKE Control Statement Format

PERMANENT FILE AUDIT REQUEST

The control statement sequence transferred by an MFLINK statement can include an AUDIT statement. To transfer the AUDIT listing to the remote host, the AUDIT statement must specify a listing file and an MFGIVE statement specifying that the listing file must follow the AUDIT statement. The AUDIT statement cannot specify the IPR listing file option. The listing file contains ASCII data with ANSI carriage control characters.

NOTE

The AUDIT statement cannot exceed 2000 characters.

The statements required for accessing a permanent file (ATTACH or PATTACH) must be executed before the MFTAKE statement if the file is permanent prior to the MFTAKE. To determine how to send a permanent file request to a CYBER 200 system, refer to the RHF documentation of the system that is to send the request.

DIRECT ACCESS FILE TRANSFERS

RHF can transfer files with random access structure. However, the ability to access the file records randomly is maintained only if the file that receives the transferred file copy is defined with a compatible random access structure.

VSOS supports random record access with the direct access file structure described in chapter 2 of this manual. When RHF transfers a copy of a VSOS direct access file to a remote system, random access of the file records on the remote system is possible only if the file that receives the file copy is defined with a compatible random access structure.

Similarly, when a random access file is transferred to a CYBER 200 system, VSOS can access the file records randomly only if the receiving file was defined with direct access file structure.

For example, suppose a file named IBM.DA.FILE is to be transferred from an IBM system to a CYBER 200 system. The file has fixed-length 80-character records, compatible with a valid VSOS direct access file structure. Assuming that the file records are to be accessed randomly on the VSOS system, a VSOS job should use the following statements to define the receiving file and transfer the file copy.

```
DEFINE,A,RLMAX=80,SFO=D.
MFLINK,A,ST=IBM,DD=C8,JCS="GET,DSN=IBM.DA.FILE".
```

Table 3-2 lists the logical structure conversions that can be specified by the data format declaration parameter (DD=) on the MFLINK control statement. Refer to chapter 4 in this manual for more information on MFLINK.

Table 3-2. Logical Structure Conversion

Permanent File Transfer DD=dd Parameter	RHF Conversion	SIL Format
UU	No logical structure conversion. RHF transfers the file as a string of bits terminated by an end-of-information protocol parameter.	All SIL record types can be read/written.
US	Logical structure indicated by file structure control words.	Only control word (W) format can be read/written.
C8	Logical structure indicated by ASCII unit separator, group separator, and file separator characters. The file contains character data from a character set with more than 64 character codes.	All SIL record types except U format can be read/written.
C6	The file contains character data from a character set with 64 or fewer character codes. Logical structure is indicated by ASCII unit separators, group separators, and file separator characters. C6 and C8 are treated identically by VSOS but may be treated differently by the remote host.	All SIL record types except U format can be read/written.
omitted	The file is treated by both the sending and the receiving host as being in the native character set of that host. Logical structure is indicated by ASCII unit separators, group separators, and file separator characters.	All SIL record types except U format can be read/written.

FILE ARCHIVING

File archiving is the process of copying permanent files to backup storage and reloading the backup copies if needed. File archiving preserves a backup copy in the event that the original copy is inadvertently destroyed. The CYBER 200 file archiving statements, DUMPF and LOADPF, are described in chapter 4 of this manual.

Using the RHF application DLF, a CYBER 200 system can archive its permanent files on a remote computer system connected via the LCN. Enter a DUMPF or LOADPF statement as described in chapter 4. The DLF application interprets the statement. It uses the following parameter specifications to communicate with the remote system.

<u>Parameter</u>	<u>Purpose</u>
ST	Specifies the remote system with which DLF communicates.
SI	Specifies the set identifier on the remote system to which the files are dumped or from which files are loaded.
JCS or INPUT	Specifies the source of the text string sent to the RHF software executing on the remote system. The text string contains the job control language required to execute the archiving request on the remote system.

TASK TERMINATION

Control task termination processing by enabling either or both of the following in the task.

- User reprieve processing
- Abnormal termination control (ATC) processing

User reprieve processing is performed when a task terminates normally or abnormally. ATC processing is performed only when the task terminates abnormally.

USER REPRIEVE

Enable user reprieve processing with a Q5REPREV call within the task.

The Q5REPREV call specifies the entry point given control when the task terminates. The entry point must be declared external within the task.

The user-defined reprieve subroutine can save information from the task whether the task executes successfully or not.

NOTE

The reprieve subroutine must return control to the system with a Q5TERM call.

If the task fails for a reason other than time limit, the reprieve routine is given the processing time remaining for the task. If the task fails because of a time limit, the reprieve routine is given an additional one-half system second.

ABNORMAL TERMINATION CONTROL

The ATC feature allows special processing to be set up if the system fails during program execution. The system failure may or may not be caused by the user's program. (The errors are listed in appendix B.) When the failure occurs, normal processing is interrupted, and all current information about the program is saved. Processing of the abnormal condition is done in interrupt mode.

ATC does not process computation errors; to process those errors, the task can call the FORTRAN Library Data Flag Branch Manager routines described in the CYBER 200 FORTRAN Reference Manual.

ATC Interrupt Subroutine

To set up interrupt processing, write an interrupt subroutine to perform the error processing that the program requires. For example, the interrupt subroutine can test the error code to determine whether the program can continue. It can also print the contents of the program variables at the time the error occurred to assist in analysis of the error.

The first line of the subroutine must have the format shown in figure 3-8. The system error codes passed to the subroutine are listed in appendix B, table B-3.

SUBROUTINE subname(errcode,pcounter,invis,regs)	
or	
ENTRY subname(errcode,pcounter,invis,regs)	
errcode	System error code (refer to table B-3).
pcounter	Virtual bit address at which the system detected the error (contents of program counter).
invis	Invisible package of interrupted task (40-word array).
regs	Register file of interrupted task (256-word array).

Figure 3-8. Interrupt Subroutine Header

You can include a Q5RFI call in the interrupt routine to return control to or to abort the interrupted task. If you omit the Q5RFI call, the task is aborted when the interrupt subroutine terminates.

These system messages can cause an interrupt deadlock with ATC, preventing completion of the task. Explicit I/O requests issued as a result of a FORTRAN statement or an SIL call within the ATC interrupt subroutine are processed correctly.

Terminal interrupts are ignored within the ATC interrupt subroutine even if the subroutine contains an SIL call or system message to process terminal interrupts. If the ATC interrupt subroutine returns control to the interrupted task, the task can then process terminal interrupts, although it might not process correctly the terminal interrupts received during ATC processing.

Enabling and Disabling ATC

Insert a Q5ENATI call in the program where abnormal termination control is to begin. On the call, specify the interrupt routine to be used. To change the interrupt subroutine used, issue another Q5ENATI call, naming another subroutine.

Insert a Q5DISATI call in the program where abnormal termination control is to end.

Abnormal termination control does not function under any of the following conditions:

- The program is already in interrupt mode.
- The program has exceeded its error recovery limit.
- The program encounters a second time limit error.

Interrupt Mode Disable

Abnormal termination control does not function if the program is already in interrupt mode. The program is in interrupt mode when it is processing a terminal interrupt, when it is performing certain I/O functions, or when it is in the abnormal termination control interrupt subroutine. If a fatal error occurs while the program is in interrupt mode, the program aborts without abnormal termination control processing.

Error Recovery Limit Disable

Abnormal termination control does not function if the program has exceeded its error recovery limit. Specify an error recovery limit (1 to 256 recoveries) on the Q5ENATI call. If an error recovery limit is not specified, the default limit of 25 recoveries is used.

Second Time Limit Disable

Abnormal termination control does not function when the program encounters a second time limit error. After encountering the first time limit, the program is given additional time for interrupt subroutine processing. (The amount of time is set by an installation parameter; it is usually 500,000 STUs.)

RESOURCE ALLOCATION

The maximum of system resources allowed for a job or task depends on the job category to which the job belongs.

A job category is identified by a one- to eight-character name. The installation defines the following limitations for jobs belonging to a job category.

- Maximum number of jobs belonging to the category that can concurrently execute
- The following limits for each job in the category:
 - Maximum and default priority
 - Maximum and default time limit
 - Maximum working set size
 - Large page limit

For further details on resource limitations imposed at security-sensitive sites, refer to chapter 7 of the Installation Handbook.

BATCH RESOURCE LIMITS

The RESOURCE control statement (refer to chapter 4 of this manual) allows specification of system resources to be used by the job. If the RESOURCE control statement is not included, the job assumes the limits specified by the JDEFAULT category.

If the RESOURCE control statement is included, the following process is used to determine resource allocation.

1. The input queue manager collects all limits supplied.
2. If the JCAT=jcat parameter is supplied, that job category is selected. The input queue manager then performs the normal limits validation for that job category.
3. If the JCAT=jcat parameter is not supplied, the input queue manager selects the job category automatically, comparing the limits provided on the RESOURCE control statement with the known limits for each job category valid for the user. All users are validated automatically for the JDEFAULT category. The category selected is based on the following criteria.
 - a. The time limit (TL), large page limit (LP), and working set size (WS) specified on the RESOURCE control statement are compared to all job category limits for these parameters. A list of eligible job categories is selected. If none of these parameters are specified, all job categories are selected.
 - b. The priority (PR) and working set size (WS) parameters are then compared to all selected job categories. If both are specified, the category with the closest fit to at least those limits is selected. If PR only is specified, the category with the minimum working set size and the closest fit to at least the specified priority is selected. If WS only is specified, the job category with the maximum priority that has the closest fit to at least the specified working set size is selected.
 - c. If neither PR nor WS is specified, the job category with the maximum priority is selected.

INTERACTIVE RESOURCE LIMITS

Within an interactive session, resource allocation is independent for each task requested. The task resource limits can be specified on the interactive execute line (refer to figure 3-2).

An interactive task is not started if a memory or time limit requested on its execute line exceeds the limit for the INTRACTV job category. If its requested priority exceeds the maximum priority for the INTRACTV job category, its priority is set at the maximum. Otherwise, its requested limits become the initial task limits.

A Q5SETLP call can change a large page limit within a task as long as the new limit does not exceed the initial task limit.

ACCOUNTING

One of the user number validations is the number of system seconds available for use by tasks belonging to the user number. A system second is one million system time units (STUs) or one million system billing units (SBUs). The site determines whether STUs or SBUs are used and the weighting factors used in the STU or SBU algorithm. For more information, refer to Accounting in volume 2 of this manual.

As tasks belonging to the user number execute, the system decrements the number of system seconds available to the user number. The system sends an error message to the job dayfile or to the interactive terminal when too few system seconds are available to perform the requested task. For a new allowance of system seconds for the user number, ask site personnel.

Statistics on jobs and tasks executed are stored in a cumulative accounting buffer and in the accounting file. The site can use the accumulated statistics to charge the user for resources used. For more information about accounting statistics, refer to Accounting in volume 2 of this manual.

Send accumulated resource usage information to the job dayfile by executing a SUMMARY statement within the job.



CONTROL STATEMENTS

The control statements described in this chapter perform job processing functions or initiate execution of system-supplied utility programs. Table 4-1 lists the control statements described in the chapter.

A control statement can be entered as either a batch execute line or an interactive execute line. The batch and interactive execute line formats are described in chapter 3 of this manual.

Table 4-1. Control Statements (Sheet 1 of 4)

Batch Job Only	
BEGIN	Insert a procedure in the control statement sequence.
COMMENT	Send a message to the job dayfile.
DAYFILE	Copy the job dayfile.
ELSE	Terminate/start control statement processing.
ENDIF	Start control statement processing.
EXIT	Set abnormal termination path.
IF	Test a condition to process or skip control statements.
NORERUN	Set norerun status.
PROC	Identify a procedure and its formal parameters.
RERUN	Set rerun status.
RESOURCE	Set job limits.
SET	Change job characteristics.
SUMMARY	Provide resource usage information.
TV	Set threshold value.
USER	Validate user access.

Table 4-1. Control Statements (Sheet 2 of 4)

System Access	
LIMITS	List user's validation controls and limitations.
PASSWORD	Change the user password.
USER	Identify the user number to which a batch input file belongs.
CHARGE	Assign account and project numbers.
File Management	
COMPARE	Compare the contents of two files.
COPY	Copy, byte by byte, the contents of one file to another.
COPYL	Copy logical partitions from one file to another.
FILES	List file information.
GIVE	Change file ownership.
LISTAC	List access permission sets.
PERMIT	Change file access permissions.
Q	List job status.
REQUEST	Create a local file, a tape file, or a file connected to a terminal.
RETURN	End file access by the job or interactive session.
SWITCH	Change file characteristics.
Queue File Management	
DIVERT	Change the destination of an output file.
DROP	Remove a job from a queue.
MFQUEUE	Submit a file to a queue on a remote system.
SUBMIT	Submit a job to the input queue on the CYBER 200.

Table 4-1. Control Statements (Sheet 3 of 4)

Permanent File Management	
ATTACH	Attach permanent files.
AUDIT	List permanent file information.
DEFINE	Create a permanent file, or make an existing local file permanent.
DMAP	Provide information on the location of permanent file segments.
DUMPF	Copy permanent files to archive storage.
LOADPF	Reload files from archive storage.
MFLINK	Transfer a permanent file between mainframes.
PURGE	Destroy permanent files.
Pool File Management	
PACCESS	Grant pool access.
PATTACH	Attach a pool.
PCREATE	Create a pool.
PDELETE	Remove pool access.
PDESTROY	Destroy a pool.
PDETACH	Detach an attached pool.
PFILES	List pool information.
Tape File Management	
BLANK	Blank label a tape volume. (For more information, refer to the VSOS Operator Guide.)
LABEL	Supply label information for a tape file.
REWIND	Rewind a tape file.
SKIP	Position a tape file.

Table 4-1. Control Statements (Sheet 4 of 4)

Code File Management	
LOAD	Create a controllee file.
OLE	Edit or create an object library.
SLGEN	Generate a shared library.
TASKATT	Alter a controllee attribute.
Debugging	
DEBUG	Debug a program (refer to chapter 6).
DUMP	Dump a drop file (refer to chapter 6).
LOOK	Dump virtual space (refer to chapter 6).
File Update	
UPDATE	Maintain a card image file (refer to chapter 5).
Privileged User Only	
EDITPUB	Add or destroy a public file.

CONTROL STATEMENT PARAMETER FORMAT

Control statement parameters use the following formats:

- keyword
- keyword=value
- value

The format used for a parameter is shown in the control statement format. Parameters that use a keyword can appear in any order within the control statement. A parameter specified only as a value must appear in its designated position, as shown in the control statement format.

Control statements recognize abbreviated keywords for many parameters. For example, the keyword DEVICE can be entered as DEVICE, DEVIC, DEVI, or DEV. The fewest characters required for keyword recognition are underlined in the parameter description.

A parameter value is passed to the utility as specified in the control statement. In general, value interpretation follows these conventions:

- Addresses are interpreted as hexadecimal constants unless indicated otherwise in the parameter description.
- Other digit strings are interpreted as decimal numbers unless a # character precedes the string. The # character indicates that the number is in hexadecimal representation. Either decimal or hexadecimal representation is allowed unless indicated otherwise in the parameter description.

Any errors in the parameters submitted are reported to the dayfile unless otherwise noted.

Standard control statement parameter processing is performed by the Q7KEYWRD routine described in volume 2 of this manual.

INTERACTIVE CONTROL STATEMENT EXECUTION

All control statements described in this chapter, except the batch processor control statements and the RESOURCE statement, can be used in an interactive terminal session. Any differences between interactive and batch use are described in the control statement description.

The syntax of a control statement entered at an interactive terminal can have either of the following formats:

- A control statement with only an optional right parenthesis or period terminator
- A control statement with one or more parameters on one or more lines

When a control statement is entered without parameters, the task responds with a prompting message, such as PLEASE SPECIFY PARAMETERS. The prompting message might also include more specific information about appropriate entries, such as a message SPECIFY: FILENAME, LENGTH, OPTIONS. In response, enter a parameter or a string of parameters separated by commas. Each entry must be terminated by a carriage return.

When the control statement is entered on a single line, the task name must be followed by a blank, a comma, or a left parenthesis. Other parameters can be separated by blanks or commas also. Depending on the utility, some parameters have subfields separated by the character slash. Any blank immediately adjacent to a parenthesis, comma, slash, or period is ignored.

Except for the LOAD control statement (described in this chapter), a control statement can be entered on more than one line. No prompting occurs between continued lines. To continue a control statement entered interactively, the character & must be the last character before the carriage return. Thus the next entry line is a continuation of the string of characters in the previous entry. Several lines can be concatenated, up to a limit of 4096 characters. If the character & is not entered, the next line is a new statement. The following entries are equivalent.

```
RETURN FILE1,FILE2,FILE3,FILE4
```

```
RETURN FILE1,FILE2,FI&  
LE3,FILE4
```

Any error in the parameters submitted and any errors encountered during execution of the utility are reported at the terminal. Almost all control statements notify you of successful execution.

CONTROL STATEMENT MANAGEMENT

You can direct the batch processor to process or skip a control statement through a control statement variable in the IF, ELSE, or ENDIF control statements. This section describes that procedure.

CONTROL STATEMENT VARIABLES

Use these variables in SET and IF statements. The SET control statement is used to place a value in a variable; refer to SET - Change Job Characteristics later in this chapter. The IF statement, described later in this chapter under IF Control Statement, is used to test control statement variable values.

The names of the global control statement variables follow:

- Rn Identifies a global variable that may be set or altered by the SET statement and referenced in an IF statement. n is an integer from 0 to 9. There is one set of these variables that exists across all control statement procedures.
- TV Returns the current threshold value; this variable cannot be set by the SET statement.
- RC Returns the last return code returned from a controllee; this variable cannot be set by the SET statement.

Following are some examples of using control statement variables:

```
SET (R0=25)
COMMENT. VARIABLE R0 NOW CONTAINS DECIMAL 25
SET(R2=#10)
COMMENT. VARIABLE R2 NOW CONTAINS HEX 10
SET(R3=TV)
COMMENT. VARIABLE R3 NOW CONTAINS THE THRESHOLD VALUE
SET(R1=R2)
COMMENT. VARIABLE R1 NOW CONTAINS HEX 10
SET(R0+=25)
COMMENT. VARIABLE R0 NOW CONTAINS DECIMAL 50
SET(R0=-10)
COMMENT. VARIABLE R0 NOW CONTAINS DECIMAL 40
SET (R7="ABC")
COMMENT. VARIABLE R7 NOW CONTAINS THE STRING "ABCxxxxx".

IF, R7="ABC".
COMMENT. THIS COMMENT STATEMENT WILL BE PROCESSED
ELSE.
COMMENT. THIS COMMENT STATEMENT WILL NOT BE PROCESSED
ENDIF.

IF, R1>R0.
COMMENT. THIS COMMENT STATEMENT WILL NOT BE PROCESSED
ELSE.
COMMENT. THIS COMMENT STATEMENT WILL BE PROCESSED
ENDIF.
```

CONDITIONAL CONTROL STATEMENTS

The statements described in this section are used to test conditions; the results of the test then cause the control statements to be processed or skipped.

IF Control Statement

IF tests control statement conditions to see if they are true or false.

If a condition that an IF control statement tests is true, the control statements that follow the condition are processed until a matching ELSE or ENDIF statement is found. The statements between the matching ELSE and ENDIF are skipped.

If the condition is false, the control statements that follow are skipped until a matching ELSE or ENDIF statement is found.

The IF statement format is shown in figure 4-1. A left parenthesis can replace the first comma in the format and a right parenthesis can replace the terminating period.

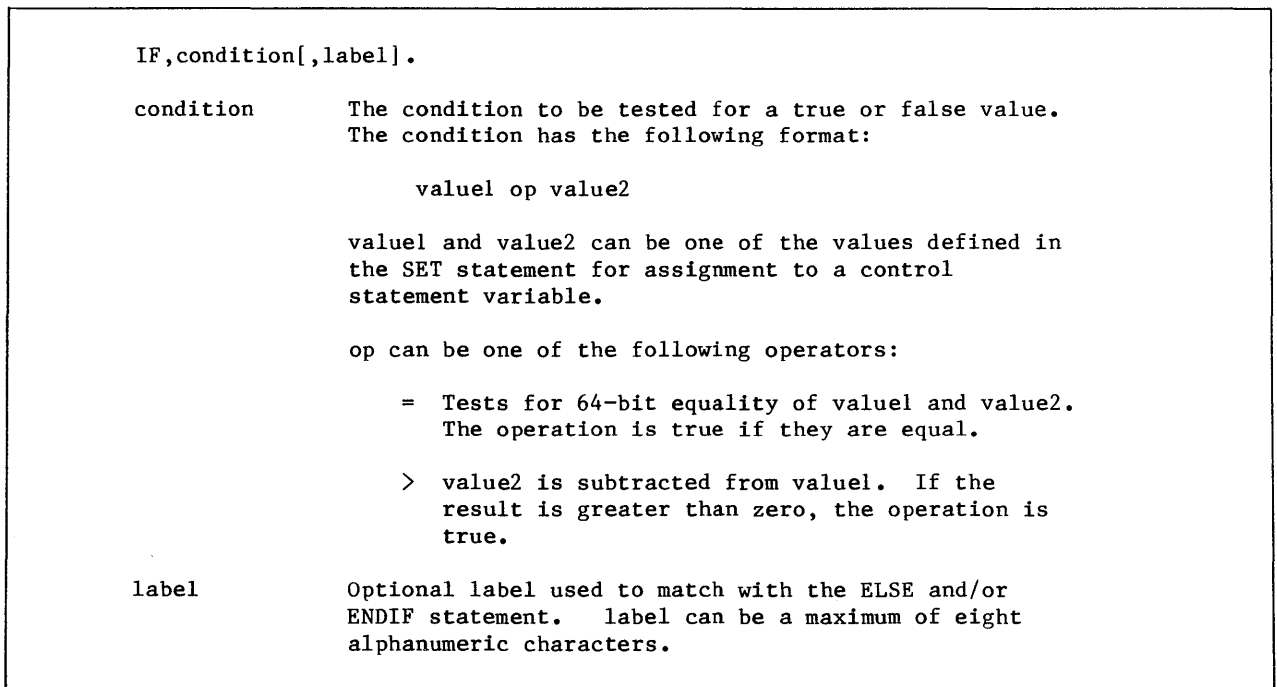


Figure 4-1. IF Control Statement Format

ELSE Control Statement

ELSE terminates true condition processing or starts false condition processing. For a true condition, ELSE causes statements to be skipped until the matching ENDIF is found. For a false condition, ELSE stops the statement skipping.

The ELSE statement format is shown in figure 4-1.1. A left parenthesis can replace the first comma and a right parenthesis can replace the terminating period.

ELSE[,label].	
label	Optional label used to match the IF and ENDIF statements.

Figure 4-1.1. ELSE Control Statement Format

ENDIF Control Statement

ENDIF terminates the skipping of control statements.

The ENDIF statement format is shown in figure 4-1.2. A left parenthesis can replace the first comma and a right parenthesis can replace the terminating period.

ENDIF[,label].	
label	Optional label used to match the IF and ELSE statements.

Figure 4-1.2. ENDIF Control Statement Format

CONDITIONAL STATEMENT PROCESSING

The IF statement causes the batch job processor to evaluate the condition. The result of the condition and label are then saved.

If the condition is false, the batch job processor scans for an ELSE or ENDIF whose label matches the saved label. When the ELSE or ENDIF is found, the batch job processor clears the condition and label and starts processing control statements again.

If the condition is true, the batch job processor clears the condition and label and starts processing control statements.

If an ELSE is found when the condition and label are cleared, the batch job processor skips control statements until an ENDIF is found that matches the ELSE.

CONTROL STATEMENT PROCEDURES

A control statement procedure is a sequence of control statements in a mass storage file. The first statement in the file must be a PROC statement. To begin execution of a procedure, a job must process a BEGIN statement.

PROC STATEMENT

The PROC control statement is the first statement in a procedure file. It identifies the name of the procedure and any formal parameters used in the procedure statements.

The PROC control statement format is shown in figure 4-1.3. A left parenthesis can replace the first comma in the format, and a right parenthesis can replace the terminating period. The parameters must appear in the order shown.

PROC,pname,fp ₁ ,fp ₂ ,...,fp _n .	
pname	Name of the procedure (one to eight letters or digits, beginning with a letter). This parameter is required.
fp _i	Optional list of up to 16 formal parameters separated by commas. Each formal parameter is a string of one to eight characters; it can contain letters, digits, and the underline character (<u> </u>). Refer to Formal Parameter Substitution in this chapter for more information.

Figure 4-1.3. PROC Control Statement Format

BEGIN STATEMENT

The BEGIN control statement initiates execution of a control statement procedure. A BEGIN statement can appear in the control statement sequence of a batch input file or in a control statement procedure; it cannot be entered at an interactive terminal.

NOTE

A site might have installed the BEGIN utility from TOOLPL which can be used interactively.

A permanent file containing a procedure must be attached before a BEGIN statement can initiate its execution.

The BEGIN statement format is shown in figure 4-2. A left parenthesis can replace the first comma in the format, and a right parenthesis can replace the terminating period. The parameters must appear in the order shown.

BEGIN,pname,pfile,p1,p2,...,Pn•

pname Name of the procedure to be executed, as defined on the PROC statement in the procedure (one to eight letters or digits, beginning with a letter). If pname is omitted, the procedure on the file specified by pfile is executed.

pfile Name of the mass storage file containing the procedure (one to eight letters or digits, beginning with a letter). If pfile is omitted, the procedure file name is assumed to be PROCFIL.

Pi Optional list of up to 16 substitution values separated by commas. A substitution value is a string of one to eight characters; it can contain letters, numbers, and the underline character (). Other characters can be included if the string is enclosed in double quote characters ("").

The editing characters ", @, and ^ are removed from the substitution value unless the user specifies two consecutive characters for each character that is to remain. For example, """"@"""" indicates that "@" is the substitution value.

The formats used to specify substitution values are described under Matching Substitution Values to Formal Parameters in this chapter. The effect of omitting substitution values is described under Omitting Substitution Values.

Figure 4-2. BEGIN Control Statement Format

CONTROL STATEMENT EXECUTION SEQUENCE

The BEGIN statement changes the control statement execution sequence. After the batch processor executes a BEGIN statement, it executes as the next statement in the control statement sequence the first control statement in the procedure specified on the BEGIN statement.

For example, the following is the control statement group in a batch input file.

```
BEGIN,MYPROC,MYFILE.  
GO.
```

The following are the statements on file MYFILE.

```
PROC,MYPROC.  
FORTRAN.  
LOAD.
```

The first statement executed is the BEGIN statement, which starts execution of the procedure.

Each statement in the procedure is executed in sequence. If no errors occur during procedure execution, the batch processor returns to the BEGIN statement that initiated execution of the procedure and executes the GO statement that follows the BEGIN statement.

If an error occurs, the batch processor searches for the next EXIT statement in the procedure or in the control statement sequence that called the procedure.

Therefore, if no errors occur, the sequence of control statement execution is BEGIN, FORTRAN, LOAD, and GO; then the job terminates normally. If an error occurs during FORTRAN task execution, the sequence of control statement execution is BEGIN, and FORTRAN; then the job terminates abnormally.

PROCEDURE NESTING

Procedures can be nested. This means that a procedure can contain one or more BEGIN statements that initiate execution of other procedures. Up to eight nesting levels are allowed; the count starts with the BEGIN statement in the batch input file.

If an error occurs in a nested procedure, causing the batch processor to search for the next EXIT statement, the search follows the sequence in which the control statements would have been executed. The search starts with the remainder of the current procedure and continues through the remainder of each procedure at each of the nested levels. It ends with the remainder of the control statement sequence in the batch input file. If it encounters a BEGIN statement during the search, it ignores the statement and does not search the specified procedure.

For example, the following are the statements in the control statement record of a batch input file and in two procedure files.

<u>Batch Input File</u>	<u>File FILE1</u>	<u>File FILE2</u>
ATTACH,FILE1.	PROC,PROC1.	PROC,PROC2.
BEGIN,PROC1,FILE1.	ATTACH,FILE2.	FTN200.
GO.	BEGIN,PROC2,FILE2.	
EXIT.	LOAD.	
DEFINE,BINARY.		

If no errors occur during execution, the following is the control statement sequence.

```
ATTACH,FILE1.  
ATTACH,FILE2.  
FTN200.  
LOAD.  
GO.
```

If an error occurs, execution continues after the EXIT statement with the DEFINE statement.

FORMAL PARAMETER SUBSTITUTION

A procedure can contain formal parameters for which values are substituted when the procedure is executed. The BEGIN statement that initiates the procedure specifies the substitution values.

Within the procedure, a string of characters is recognized as a formal parameter if the string matches a formal parameter specified on the PROC statement and if it has a valid delimiting condition before it and after it. The following are the valid delimiting conditions.

- A character that is not A through Z, 0 through 9, or the underline character ()
- The beginning or end of the statement

For example, in the following statement the character strings that could be specified as formal parameters on a PROC statement are underlined.

```
LOAD,BINARY,CN=GO/#102.
```

Matching Substitution Values to Formal Parameters

Specify substitution values on the BEGIN statement using either or both of the following parameter formats:

<u>Format</u>	<u>Example</u>
formal parameter=substitution value	KEY1=MYFILE
substitution value	MYFILE

When both parameter formats are used on the BEGIN statement, all parameters specified as only the substitution value must precede all parameters specified as formal parameter=substitution value. This is because changing from one format to the other changes the method used to match substitution values to formal parameters.

When the first substitution value on the BEGIN statement is specified in the format for substitution value only, the batch processor matches the substitution value to the first formal parameter on the PROC statement. It continues to match substitution values to formal parameters according to their position in the parameter sequence (first with first, second with second, and so on) as long as the format for substitution value only is used.

When the batch processor encounters a BEGIN statement parameter that uses the formal parameter=substitution value format, it no longer uses positional parameter matching for the procedure. It matches substitution values to formal parameters according to their pairing on the BEGIN statement.

When using the formal parameter=substitution value format, specify parameters in any order. In this case, the parameters are not dependent on order; matching is by formal parameter instead of by sequential order.

For example, a procedure on file MYFILE has the following statements:

```
PROC,MYPROC,IN,OUT.  
COPY,IN,OUT.
```

Each of the following BEGIN statements substitutes A for IN and B for OUT in the procedure.

```
BEGIN,MYPROC,MYFILE,A,B.  
BEGIN,MYPROC,MYFILE,IN=A,OUT=B.  
BEGIN,MYPROC,MYFILE,OUT=B,IN=A.  
BEGIN,MYPROC,MYFILE,A,OUT=B.
```

The following BEGIN statement is invalid because the batch processor cannot return to the positional matching method after it uses the other method.

```
BEGIN,MYPROC,MYFILE,IN=A,B.
```

The following BEGIN statement is invalid because more substitution values are specified than formal parameters on the PROC statement.

```
BEGIN,MYPROC,MYFILE,A,B,C.
```

The following BEGIN statement is invalid because it specifies a substitution value twice for the same formal parameter.

```
BEGIN,MYPROC,MYFILE,A,IN=A,OUT=B.
```

Omitting Substitution Values

If the BEGIN statement does not specify a substitution value for a formal parameter specified on the PROC statement, the batch processor does not alter the occurrences of that formal parameter in the procedure. However, the batch processor still removes editing characters as described under Suppressing Formal Parameter Substitution and Concatenating Substitution Values in this chapter.

When matching substitution values to formal parameters according to their position in the parameter sequence, the batch processor assumes that a substitution value is omitted when it encounters either of the following in the BEGIN statement.

- Two consecutive commas
- Fewer substitution values than required for all formal parameters on the PROC statement

When matching substitution values to formal parameters according to keyword=substitution value parameters on the BEGIN statement, the batch processor assumes parameter omission if both of the following conditions exist.

- When matching substitution values to formal parameters according to position, it did not match a value with the formal parameter.
- No BEGIN statement parameter exists that equates the formal parameter to a substitution value (formal parameter=substitution value).

For example, a procedure on file MYFILE has the following statements:

```
PROC,MYPROC,COMPILE,GO,LIB1.  
LOAD,COMPILE,CN=GO,LIB=LIB1.
```

The following statements omit substitution values for formal parameters COMPILE and LIB1.

```
BEGIN,MYPROC,,BIN.  
BEGIN,MYPROC,GO=BIN.  
BEGIN,MYPROC,,GO=BIN.
```

Each BEGIN statement results in execution of the following statement:

```
LOAD,COMPILE,CN=BIN,LIB=LIB1.
```

Suppressing Formal Parameter Substitution

Prevent value substitution for an occurrence of a formal parameter by prefixing the formal parameter occurrence with the @ character. The batch processor removes the @ character before the statement is executed, but the formal parameter occurrence remains unchanged.

For example, file MYFILE contains the following statements:

```
PROC,MYPROC,OUTPUT.  
LOAD,FILE1,@OUTPUT=OUTPUT.
```

The following statement initiates execution of the procedure.

```
BEGIN,MYPROC,MYFILE,MYOUTPUT.
```

After formal parameter substitution, the following statement is executed.

```
LOAD,FILE1,OUTPUT=MYOUTPUT.
```

Concatenating Substitution Values

Concatenate substitution values after the values replace formal parameters in a procedure. Separate the formal parameters with the character ^. After the batch processor substitutes values for the formal parameters, it removes the ^ character, thereby concatenating the substitution values.

For example, file MYFILE contains the following statements:

```
PROC,MYPROC,R,W,X.  
DEFINE,FILE1,ACS=R^W^X.
```

The following statement initiates execution of the procedure.

```
BEGIN,MYPROC,MYFILE,W=A.
```

After formal parameter substitution, the following statement is executed.

```
DEFINE,FILE1,ACS=RAX.
```

The following example shows use of concatenation to specify a load address longer than eight characters. The following are the statements on the procedure file MYFILE.

```
PROC,MYPROC,P1,P2.  
LOAD,F1,0=#P1^P2.
```

The following is the BEGIN statement used.

```
BEGIN,MYPROC,MYFILE,C000,00400000.
```

The following is the LOAD statement executed.

```
LOAD,F1,0=#C00000400000.
```

Suppressing @ or ^ Character Removal

You can prevent removal of the @ or ^ character from a procedure statement when the statement is executed. To do so, you must specify two consecutive @ or ^ characters for each @ or ^ character to remain in the statement.

The batch processor groups multiple @ or ^ characters in pairs. For each pair, one @ or ^ character remains in the statement.

If an odd @ character remains after the consecutive @ characters are grouped in pairs and the consecutive @ characters are the prefix of a formal parameter, the odd @ character prevents value substitution.

The effect of the ^ character remains the same whether an odd or even number of consecutive characters are specified.

For example, assume the following conditions:

- PARM is a formal parameter.
- SV is the substitution value for the formal parameter PARM.
- NOTPARM is in the procedure but is not a formal parameter.

The following shows the effect of the @ character in the procedure.

<u>Before Substitution</u>	<u>After Substitution</u>
@PARM	PARM
@@PARM	@SV
@@@PARM	@PARM
@@@@PARM	@@SV
@NOTPARM	NOTPARM
@@NOTPARM	@NOTPARM
@@@NOTPARM	@NOTPARM
@@@@NOTPARM	@@NOTPARM

The following shows the effect of the ^ character in the procedure:

<u>Before Substitution</u>	<u>After Substitution</u>
PARMPARM	PARMPARM
PARM^PARM	SVSV
PARM^^PARM	SV^SV
PARM^^^PARM	SV^SV
NOTPARM^NOTPARM	NOTPARMNOTPARM
NOTPARM^^ NOTPARM	NOTPARM^NOTPARM
NOTPARM^^^NOTPARM	NOTPARM^NOTPARM

ATTACH - ATTACH PERMANENT FILES

The ATTACH control statement (refer to figure 4-3) attaches private permanent files.

A permanent file must be attached before it can be accessed. The permanent files are attached to the JDN of the job or interactive session in which the ATTACH statement is executed.

An ATTACH statement can attach any of the following sets of private permanent files.

- All files belonging to you
- The specified files belonging to you
- The specified files accessible to you, but belonging to another user
- Files that have segments unavailable because a device is down

An ATTACH statement cannot attach files whose security level is greater than the security level of the job or interactive session.

The ATTACH statement can specify the access modes allowed while each specified file is attached. You must have the corresponding access permission to the file for each of the specified access modes. Requests to open the file while it is attached can specify one or more of the access modes specified on the ATTACH statement.

The WAIT parameter determines whether ATTACH waits for a file that is unavailable because another job has the file attached such that file access cannot be shared (refer to Concurrent File Access in chapter 2 of this manual).

If WAIT=N is specified, ATTACH does not wait. If WAIT=Y is specified, ATTACH waits until the file is available or until it exceeds its wait time limit. (The wait time limit is specified by an installation parameter.) If the ATTACH statement is issued interactively, it displays a message while it waits for a file.

Format for Attaching Files the User Owns

```
ATTACH, { lfn-list  
          * } ,ACCESS=acs,WAIT=x,TRUNCATED=x.
```

Format for Attaching Files the User Does Not Own

```
ATTACH,lfn-list,USER=userno,ACCESS=acs,WAIT=x,TRUNCATED=x.
```

lfn-list Files to be attached. This parameter is required.
*

lfn-list List of files (1 through 16 file names separated by
commas).

* All private permanent files belonging to you. * is
mutually exclusive with USER=userno and ACCESS=acs.

USER=userno User number that owns the specified files. If USER=userno is
omitted, the user number of the job or interactive session is
used. USER=userno and * are mutually exclusive.

Figure 4-3. ATTACH Control Statement Format (Sheet 1 of 2)

ACCESS=acs Access modes allowed while the specified files are attached. The set of access modes is indicated by a string composed of one or more of the following letters:

R	Read access
W	Write access
X	Execute access
A	Append access
M	Modify access

If ACCESS=acs is omitted, only read and execute modes are allowed during the attach. ACCESS=acs and * are mutually exclusive.

WAIT=x Indicates whether ATTACH waits if a file is currently unavailable. If WAIT=x is omitted, ATTACH waits.

Y	ATTACH waits
N	ATTACH does not wait

TRUNCATED=x Allows you to access files that have segments unavailable because a device is down.

Y	Access in read mode only. The ACCESS=R parameter must be specified so that truncated files cannot be written on or executed.
N	Access is not available to truncated files. N is the default.

Figure 4-3. ATTACH Control Statement Format (Sheet 2 of 2)

ATTACH cannot attach a file if the file name is the same as that of a private file (local or permanent) that is already attached to the job.

ATTACH attaches the files in the order specified. If it cannot attach a file, it continues processing with the next file in the list. It sends a message to the job dayfile or the interactive terminal if a file cannot be attached. It does not send a message when it successfully attaches all files.

AUDIT - LIST FILE INFORMATION

The AUDIT control statement lists information about permanent mass storage files.

A privileged user can list information about any permanent file on the system. A master user can list information about all private, pool, or public files with the account identifier(s) assigned. A system user can list information about the I/O queues. A nonprivileged, nonmaster user is allowed to list information about only the files that he or she owns.

To list information about files belonging to a pool, a nonprivileged user must attach the pool before executing the AUDIT statement; a privileged user need not attach the pool. Private files can be attached or unattached.

The format of the AUDIT control statement is shown in figure 4-4. All parameters are optional and can, except for the first, appear in any order. The first parameter, if specified, must be a list of file names.

AUDIT, lfn-list, USER=userno, POOL=plist, ACCOUNT=alist, JCAT=jcatlist, LID=lidlist, MPN=mlist, DSET=devset, PACK=packlist, SELECT=opts, DATE=mmddyy, TIME=hhmm, LO=x, LIST=lfn/len.

lfn-list List of 1 through 128 file names, separated by commas. The specified files are assumed to belong to the user number specified by the USER parameter or to the pools specified by the POOL parameter. If omitted, all files that belong to userno or plist are listed. If SEL=0 is specified, the lfn-list identifies the last-group-files(s) of the output-file-family(s). If SEL=0 is specified and lfn-list is omitted, AUDIT lists information about all output-file-families.

USER=userno File owners.

For a nonprivileged user:

userno User number of the nonprivileged user.

0 Public files.

For a nonprivileged user with master user status:

u-list List of 1 through 128 user numbers separated by commas. If user number 0 is specified, AUDIT lists public file information.

* All file owners, private, pool, and public.

NOTE

The user must use the ACCOUNT parameter and must have master user status for all accounts specified.

Figure 4-4. AUDIT Control Statement Format (Sheet 1 of 4)

For a privileged user:

u-list List of 1 through 128 user numbers separated by commas. If user number 0 is specified, AUDIT lists public file information.

* All file owners, private, pool, and public.

For a system user who has specified SEL=I or SEL=O:

u-list List of 1 through 128 user number(s) that queued the files (original owner). If omitted, AUDIT lists information about all queue files.

If both USER=userno and POOL=plist are omitted, AUDIT lists information for files belonging to the user number under which AUDIT is executing.

POOL=plist List of 1 through 128 pool names separated by commas. The pools must be attached for a nonprivileged user.

ACCOUNT=alist For a nonprivileged user, alist is a list of one to seven account identifiers separated by commas. The user must be validated for all specified account identifiers. For a privileged user, alist is a list of 1 through 128 account identifiers separated by commas.

If this parameter is omitted, files are candidates for listing regardless of their account identifiers.

JCAT=jcatlist List of 1 through 64 job categories separated by commas. This parameter is allowed only if SEL=I is specified and applies only to the input queue. If SEL=I is specified and JCAT is omitted, all input queue files are listed regardless of job categories.

LID=lidlist List of 1 through 128 destination LIDs for input or output files. This parameter is allowed only if SEL=I or SEL=O is specified and applies only to the input or output queues. If SEL=I or SEL=O is specified and LID is omitted, all input/output queue files are listed regardless of their destination LIDs.

MPN=mlist List of 1 through 128 master project numbers separated by commas. If this parameter is omitted, files are candidates for listing regardless of their master project number.

DSET=devset List of 1 through 128 device sets separated by commas. Allows files to be audited on a device set basis. devset is the device set name (DVSTnn). If this parameter is omitted, all sets are audited.

PACK=packlist Allows files to be audited on a pack basis. packlist is a list of 1 through 128 pack names (PACKnn) separated by commas. Only those files beginning on a specified pack are audited. Those that merely continue from another pack are not audited.

Figure 4-4. AUDIT Control Statement Format (Sheet 2 of 4)

<u>SELECT=opts</u>	<p>File characteristics of all files audited (any combination, except as noted, of the following letters without separators). A file must meet all characteristics specified, in order, to be audited.</p> <p>A Files accessed on or after the date and time specified by the DATE and TIME parameters.</p> <p>C Files created on or after the date and time specified by the DATE and TIME parameters.</p> <p>I Files in the input queue. Only the system user can select this option. The I option is mutually exclusive with the PO and PROJ parameters.</p> <p>M Files modified on or after the date and time specified by the DATE and TIME parameters.</p> <p>N Reverses the meaning of all the A, C, or M options; that is, NA means not accessed, NC means not created, and NM means not modified. ANCM means not accessed, not created, and not modified.</p> <p>O Files in the output queue. Only the system user can select this option. The O option is mutually exclusive with the PO and PROJ parameters.</p> <p>X Files expired. A file expires when the current date is greater than the file creation date plus its retention period.</p> <p>If SELECT=opts is omitted, AUDIT assumes no options.</p>
<u>DATE=mmddy</u>	<p>Date used by the A, C, and M options on the SELECT=opts parameter. The first two digits of the date indicate the month, the next two digits the day of the month, and the last two digits the last two digits of the year.</p> <p>If DATE=mmddy is omitted, AUDIT uses the current date.</p>
<u>TIME=hhmm</u>	<p>Time used by the A, C, and M options on the SELECT=opts parameter. hh is the hour, based on a 24-hour clock. mm is the minute in the hour.</p> <p>If TIME=hhmm is omitted, AUDIT uses midnight.</p>

Figure 4-4. AUDIT Control Statement Format (Sheet 3 of 4)

LO=x

Audit information required:

F Full audit.

P Partial audit.

If LO=x is omitted, AUDIT writes partial audit information.

LIST=lfn/len

Listing file specifications:

lfn File name (one to eight letters or digits, beginning with a letter). If lfn is omitted, AUDIT uses file OUTPUT.

len File length in 512-word blocks. If len is omitted, the file length is #40 blocks.

Figure 4-4. AUDIT Control Statement Format (Sheet 4 of 4)

FILE SPECIFICATION

The set of files for which AUDIT lists information can be specified by name or by attributes. The set of files must have all the attributes specified. The USER and POOL parameters specify file ownership, the DSET and PACK parameters specify file residence, and the SELECT, DATE, and TIME parameters can specify file usage and age.

If no file names are specified and the USER and POOL parameters are omitted, AUDIT lists information about permanent files belonging to the user number for which AUDIT is executed.

Table 4-2 summarizes the interaction of the USER and POOL parameters.

Table 4-2. Interaction of USER and POOL Parameters for AUDIT, DUMPF, and LOADPF

Files Processed	Privileged User						Nonprivileged User			
	No USER		USER=list		USER=ALL		No USER		USER=userno†	
	No POOL	POOL=plist	No POOL	POOL=plist	No POOL	POOL=plist	No POOL	POOL=plist	No POOL	POOL=plist
User private files	X						X			
Listed user private files (or public files if user number 0 is specified)			X	X					X	X
Listed pool files		X		X				X		X
All files regardless of owner (including public and pool files)					X	X				

†For AUDIT, a nonprivileged user can specify his or her own user number or 0, the public file user number.

AUDIT OUTPUT

The LO parameter on the AUDIT control statement determines whether AUDIT produces a full or partial output listing. A full listing produces all of the headings described next (with the exception of ACCOUNT and MPN), while a partial listing contains only the first 12 headings. A full listing does not exceed 132 characters, excluding the carriage return, and a partial listing does not exceed 80 characters, excluding the carriage return. Dates appear as month, day, year. Time appears in a 24-hour format. All values are decimal unless noted otherwise.

AUDIT prints a report of each account identifier, starting at the top of a new page. Print lines do not exceed 80 characters for partial listings and 132 characters for full listings, excluding carriage return. Dates appear as month, day, and year; time appears as a 24-hour clock.

The following is a list of the column headings used in a full AUDIT listing and the information given under each heading.

<u>Heading</u>	<u>Description</u>
FSN	File sequence number. Hexadecimal count of files audited.
NAME	File name. The file name is suffixed with an asterisk (*) if the file is a production file.
OWNER	File owner: individual user number, public user number (0), or pool name. If SEL=I or SEL=O is specified, then the user number of the original file owner is listed.
TYP	File type: controllee or data [virtual code (VC) or physical data (PD)].
FC	File category: batch input file (B), input queue file (I), output queue file (O), user file (U), system-generated drop file (S), or not defined (N).
RT	Record type: ANSI fixed length (F), record mark delimited (R), undefined (U), or control word (W).
BT	Blocking type: character count (C), internal (I), or record count (K).

<u>Heading</u>	<u>Description</u>
ACS	Access permission set: read (R), write (W), execute (X), append (A), modify (M) permissions, no permissions (NONE), or purge only (PURGE). AUDIT lists the owner's access permission set for private files and the general access permission set for pool and public files.
EXT	File allocation: segmentable (S) and/or extendable (X).
SL	Security level: 1 through 8.
DEVICE	Device name of mass storage file. An asterisk following the device name indicates that a portion of the file resides on another disk.
DSET	Name of device set.
FLEN	Number of 512-word blocks in file.
FACT	Account identifier.
DORG	Creation date (date of origin).
TORG	Creation time (time of origin).
DOLA	Date of last file access.
TLR	Time of last file access.
DOLM	Date of last file modification.
TOLM	Time of last file modification.
EXP	Expiration date (creation date plus retention period).
ACCOUNT	Account identifier.
MPN	Master project number.

If SEL=I or SEL=O is specified, the TYP column is deleted and the following column headings replace ACS and EXT:

<u>Heading</u>	<u>Description</u>
LID	Destination lid for output and/or input queue files.
JCAT	Job category of input queue files. For all other file types, this field is left blank.

Figure 4-5 shows an example of a partial AUDIT output listing as produced by the following control statement:

```
AUDIT,U=0,040018,LO=P.
```

FSN	NAME	OWNER	TYP	FC	RT	BT	ACS	EXT	SL	DEVICE	DSET	FLEN
1	POOLFILI	40018	PD	U	W	C	RX	SX	1	PACK01	DVST01	4321
2	EXECFIL	0	VC	U	W	C	X	SX	1	PACK01	DVST01	22
3	MYFILE	40018	PD	U	W	C	RX	SX	1	PACK1F	DVST14	123
4	NEWFILE	0	PD	U	W	C	MAR	SX	1	PACK30	DVST1E	16
5	PUBFILE	0	PD	U	W	C	MAR	SX	1	PACK20	DVST14	148000

Figure 4-5. AUDIT Output Example

If the ACCOUNT and/or MPN parameters are specified on the AUDIT control statement, files are sorted by account, master project number, user number, and file name. For every combination of ACCOUNT and MPN, a subheader and subtotals are printed on the AUDIT output.

Figure 4-6 shows an example of an AUDIT output listing as produced by the following control statement:

```
AUDIT,AC=AAAAAA,MPN=AUD,DEF
```

CYBER 200 AUDIT -				USER 112311				05/14/86		06.01.32		PAGE 1	
ACCOUNT = AAAAAA				MPN = AUD									
FSN	NAME	OWNER	TYP	FC	RT	BT	ACS	EXT	SL	DEVICE	DSET	FLEN	
1	TRETST	112311	PD	U	R	C	XMARW	X	1	PACK4F	DVST1F	4	
TOTAL SIZE =										4			
ACCOUNT = AAAAAA				MPN =DEF									
FSN	NAME	OWNER	TYP	FC	RT	BT	ACS	EXT	SL	DEVICE	DSET	FLEN	
1	ONETST	112311	PD	U	R	C	XMARW	X	1	PACK1F	DVST1F	16	
2	TWOTST	112311	PD	U	R	C	XMARW	X	1	PACK1F	DVST1F	16	
TOTAL SIZE =										32			

Figure 4-6. AUDIT Output Example
(if either the ACCOUNT or the MPN parameters are specified)

CHARGE - ASSIGN ACCOUNT AND PROJECT NUMBER

The CHARGE control statement allows you to change the account identifier and change or define a project number within a batch job, an interactive session, or PTF processing. The account identifier specified on the CHARGE statement must be valid for the user number under which the statement is being executed.

The CHARGE statement can be issued several times during a batch job, interactive session, or PTFS processing. There is no limit to the number of times the statement can appear. The CHARGE statement remains in effect until either a new CHARGE statement is issued or the job terminates. When a new CHARGE statement is processed, the accumulated SBU/STU information is written to the user and system dayfiles and to the account file. (SBU/STU information is not written to the user dayfile for interactive sessions or PTFS processing.) Subsequent job processing is charged to the newly supplied charge number.

There are two SBU/STU accumulators: one to accumulate SBUs/STUs for the job and the other to accumulate SBUs/STUs for the current account identifier and project number. Both of the SBU/STU totals are printed in the dayfile and the account file at the end of a batch job if a project number exists; otherwise only the job SBU/STUs are written to the dayfile.

You, like all users, have a default account identifier assigned to your user number. You also have the option of being validated with a default project number via validations. At the start of a batch job or interactive session, if you do not issue a CHARGE statement, the default project number, if one exists, and the account identifier specified on the USER or LOGIN statement or the default account identifier are assigned to the job or session. If a CHARGE statement is entered, both the account identifier and the project number must be specified on the statement.

If your user number has user project control (UPC) status and no project number assigned as default, you must supply a CHARGE statement as the first executable statement in the batch job, interactive session, or PTFS processing. If you do not supply a CHARGE statement as the first executable statement after the RESOURCE statement in a batch job, the job is aborted with no exit processing. For an interactive session, no other statements are accepted until the CHARGE statement is executed. For the transferring of permanent files to the CYBER 205 from a remote host through PTFS, you must supply a CHARGE statement after the USER statement if you have UPC status and no project number; otherwise, the transfer aborts.

The master project number (the first one to three characters of the project number, excluding the special characters * and -) is appended to the account identifier and assigned to any mass storage files created during the job or session.

The following are examples of master project numbers extracted from project numbers.

<u>Project Number</u>	<u>Master Project Number</u>
ABCDEF	ABC
A-B*CDEF	ABC
A*B*	AB
*ABC-DEF	ABC

Figure 4-7 shows the CHARGE control statement format.

CHARGE,account,project.	
account	CYBER 200 account identifier (one to eight alphanumeric characters). This parameter is positional and mandatory.
project	A project number (1 to 20 alphanumeric characters, including the special characters * and -). This parameter is positional and mandatory.

Figure 4-7. CHARGE Control Statement Format

COMMENT - SEND MESSAGE TO JOB DAYFILE

The COMMENT or * control statement enters a message in the job dayfile.

A COMMENT or * statement is only valid within a batch job. The batch processor executes the COMMENT or * statement itself; it does not initiate execution of a utility program.

Figure 4-8 shows the COMMENT and * control statement format. A period is required after COMMENT, but no space need appear after the required period; no ending punctuation is needed at the end of the message. Multiple COMMENT or * control statements are required to send a message longer than the number of columns available on a single execute line.

COMMENT.message	
* message	
message	String of characters to be sent to the job dayfile. The string should contain only characters that can be printed at a line printer.

Figure 4-8. COMMENT and * Control Statement Format

COMPARE - COMPARE FILE CONTENTS

The COMPARE control statement compares, bit for bit, the contents of one file with that of another. If the contents of the two files do not match, COMPARE lists the contents of the nonmatching words.

COMPARE can compare mass storage files and tape files. Compared mass storage files can be attached permanent files or temporary files; they can be data files or controllee files. COMPARE starts its comparison on both files at the beginning of information plus any relative word offset.

COMPARE does not detect whether files that have been blank compressed which are otherwise equivalent when expanded are equivalent if the files were blank compressed using different formulas.

If a compared file is a labeled tape file, COMPARE compares the contents of the file from its HDRI label to its EOFI label; it does not compare the contents of the file labels. If a compared file is an unlabeled tape file, COMPARE compares the contents of the file from load point to the double tape marks or EOFI label that marks the end of the file.

The compared files should have the same record format. COMPARE reads the contents of each file as a continuous bit string, not as a sequence of records. Therefore, it compares record delimiters within the data.

COMPARE lists the nonmatching words in the dayfile of a batch job or at the terminal of an interactive session.

Figure 4-9 shows the COMPARE control statement format. Only the first two parameters, which specify the compared files, are required.

COMPARE,alfn,blfn,L=len,A=aadr,B=badr,N=lt.	
alfn,blfn	Names of the files to be compared.
<u>L</u> =len	Hexadecimal number of words to be compared. If the L parameter is omitted, comparison stops at the end of the shorter file.
<u>A</u> =aadr	Relative hexadecimal word address in file alfn at which comparison is to begin, counting the first word of the file as 0. If the A parameter is omitted, comparison begins with the first word of file alfn.
<u>B</u> =badr	Relative hexadecimal word address in file blfn at which comparison is to begin, counting the first word of the file as 0. If the B parameter is omitted, comparison begins with the first word of file blfn.
<u>N</u> =lt	Decimal number of nonmatching words allowed before comparison stops. Both the nonmatching words and their relative locations are displayed. If N=lt is omitted, comparison stops at the first nonmatching word.

Figure 4-9. COMPARE Control Statement Format

CONTROLLEE FILE COMPARISON

When comparing controllee files, you might not want to compare the minus page and register file stored in the first two blocks of the files. To omit the first two 512-word blocks from the comparison, specify the starting compare address for each file with the A and B parameters. The starting compare addresses are specified as hexadecimal word addresses.

For example, the following statement omits the first two blocks from the comparison of files FILE1 and FILE2.

```
COMPARE,FILE1,FILE2,A=400,B=400.
```

COPY - COPY A FILE

The COPY control statement copies the contents of one file to another file.

The file copied (the input file) must be an existing file. It can be a mass storage file or a tape file. The file copied (the output file) can also be a mass storage file or a tape file, but it need not exist unless it is a tape file. COPY first searches for the local and attached permanent files and tape files for the specified files. Then, if necessary, it searches the attached pools. COPY cannot copy to public files or system pool files.

If the file does not exist, COPY creates a local mass storage file. The new file is created with the name specified for the output file and the same characteristics as the input file, including its type, access permissions, record format, internal characteristics, and length.

NOTE

If the original file is a system-created drop file, the output file is a user-created drop file.

By default, the copy begins at the beginning of the input file. However, a different starting location for the input file may be specified with the I parameter and a different starting location for the output file with the O parameter.

By default, the copy ends when COPY encounters the end of the input file. However, if the L parameter is specified, the copy ends when COPY has copied the specified number of words.

COPY returns status and error information to the dayfile of a batch job or to the terminal for an interactive session. It always displays the number of words copied (in hexadecimal).

In a batch job, the COPY control statement cannot copy to a file by the name of INPUT. If this is attempted, the COPY utility issues an error message that prohibits the file from being copied to file INPUT.

Figure 4-10 shows the COPY control statement format. The first two parameters specifying the input and output files are required and must appear in the order shown.

```
COPY,inlfn,outlfn,L=len,I=inadr,O=outadr,PACK=packid,IRW=irw.
```

inlfn Name of file to be copied.

outlfn Name of file that contains a copy of all or part of file inlfn. It can be either an existing file or a new file COPY creates.

L=len Hexadecimal number of words to be copied.

I=inadr Relative hexadecimal word address in file inlfn at which copying is to begin, counting the first word of the file as 0.

If the I parameter is omitted, inlfn is copied from its beginning.

Figure 4-10. COPY Control Statement Format (Sheet 1 of 2)

<u>O</u> =outadr	Relative hexadecimal word address in file outlfn at which copied information is to be placed, counting the first word of the file as 0. If the O parameter is omitted, the copy begins at the beginning of outlfn.				
<u>PACK</u> =packid	Identifier for a pack in the device set on which outlfn is to reside. If outlfn already exists on another pack, the system ignores this parameter, copies inlfn to the existing outlfn, and sends a warning message to the job dayfile or interactive terminal. If packid is omitted and outlfn does not exist, the system selects a pack and creates outlfn.				
<u>IRW</u> =irw	Inhibit rewind of input and output files. This applies to tape files only. <table border="0" style="margin-left: 40px;"> <tr> <td style="text-align: center;"><u>Y</u></td> <td>Files are not rewound prior to copying.</td> </tr> <tr> <td style="text-align: center;"><u>N</u></td> <td>Files are rewound prior to copying.</td> </tr> </table> <p>If IRW=irw is omitted, N is used.</p>	<u>Y</u>	Files are not rewound prior to copying.	<u>N</u>	Files are rewound prior to copying.
<u>Y</u>	Files are not rewound prior to copying.				
<u>N</u>	Files are rewound prior to copying.				

Figure 4-10. COPY Control Statement Format (Sheet 2 of 2)

COPYING TO OR FROM A TAPE FILE

To copy data to or from a tape file, execute the appropriate REQUEST and LABEL statements before executing the COPY statement. If the tape file is the input file, specify read access for the file. If the tape file is the output file, specify write access for the file.

If a labeled tape file is the input file, COPY opens the tape file and verifies its HDR1 label. If a labeled tape file is the output file, COPY opens the tape file and verifies or writes its HDR1 label, depending on the label processing option on the LABEL statement for the file.

When the copy terminates, COPY closes the tape file. The end-of-file indicator is written as required for the tape format used.

COPYING TO A MASS STORAGE FILE

When a mass storage file is specified as the output file, the output file need not be as long as the source file, unless the output file has been requested with the noextend option. COPY extends the output file as necessary. An output file that COPY creates has the same length as the input file.

However, if the O parameter is specified so that the copy does not begin at the beginning of the output file, the data in the input file does not fit in the output file.

For example, assuming FILE1 is a mass storage file, the following statement returns an error because the file created is not long enough for the data to be copied.

```
COPY,FILE1,,0=100.
```

When a tape file is copied to a mass storage file, the mass storage file is created with a length of 512 blocks if it did not exist previously.

CONTROLLEE FILE COPY

The first two blocks of a controllee file contain its minus page and register file. To omit copying the minus page and register file of a controllee file, specify the starting copy address with the I parameter. The copy address is specified as a hexadecimal word address. For example, the following COPY statement omits copying the first two 512-word blocks. }

```
COPY,FILE1,FILE2,I=400. }
```

COPYL - COPY LOGICAL RECORDS

The COPYL control statement copies logical partitions of files. Unlike the COPY statement, COPYL does record type conversion and replication of logical subsets of data files, and it can interface to files connected to terminals (files created by REQUEST statements that specify DEV=TE).

The input file must be attached or local. All device types are treated alike in positioning operations, always starting at the beginning of information.

If the output file is already a local file or attached permanent file, COPYL does not reduce the file after completing the copy. However, if COPYL creates the file, it does reduce the file after completing the copy.

If the output file is a file connected to a terminal, COPYL displays the first 24 records at the terminal; it then displays the following prompt:

```
ENTER "C" TO CONTINUE "END" TO TERMINATE
```

If you enter C, COPYL displays the next 24 records and then repeats the prompt. The display and prompt cycle repeats until you enter END or until the specified copy completes.

The value of the PART parameter determines not only the level of skip and copy, but also the structure of the output file. Input and output files are always read and written at the record level, but the output data is guaranteed to be delimited at whatever partition level was specified. If a partition delimiter with a level higher than that specified on the execute line is read on either the input to the copy operation or a skip operation, the operation is stopped; if a partition delimiter of the specified level does not already exist in the outfile, one is inserted at this point.

Before beginning the copy, COPYL skips forward the number of partitions specified by the ISKIP and OSKIP parameters. The ISKIP parameter specifies the number of partitions skipped on the input file; the OSKIP parameter specifies the number of partitions skipped on the output file.

In a batch job, the COPY control statement cannot copy to a file by the name of INPUT. If this is attempted, the COPY utility issues an error message that prohibits the file from being copied to file INPUT.

The copy terminates when the specified number of partitions have been copied or COPYL encounters a higher-level partition boundary. For example, a record copy terminates if COPYL encounters a group delimiter.

Figure 4-11 shows the COPYL control statement format.

```
COPYL,infile,outfile/len,PARTITION=part,NUMBER=number,ISKIP=ipnum,OSKIP=opnum,IRW=irw,  
NOCOMP.
```

infile Input file. The parameter is positional and required. The
 input file must be different from the output file.

outfile Output file. The parameter is positional and required. The
 input file must be different from the output file.

Figure 4-11. COPYL Control Statement Format (Sheet 1 of 2)

len Length at which outfile is created by COPYL. If infile is on disk, len defaults to the length of infile. If infile is on tape or connected to a terminal, len defaults to 128 blocks. If outline is already a local or attached permanent file, len is ignored.

PARTITION=part Indicates whether COPYL copies records or groups.

R Record

G Group

If PARTITION=part is omitted, COPYL copies records.

NUMBER=number * (all) or number of partitions (as specified by the part parameter or default) to be copied. Default is *. If a higher-level partition delimiter or an end-of-information is encountered before num is exhausted, the copy operation terminates. The normal termination message indicates the amount of data copied. If NUMBER=number is omitted when copying records, COPYL copies until it encounters a group delimiter or the end of the file. If NUMBER=number is omitted when copying groups, COPYL copies until it encounters the end of the file.

ISKIP=ipnum Number of partitions (as specified by part or default) to be skipped on infile before the copy operation is begun; default is 0. If a nonzero value is specified and a higher-level partition delimiter or an end-of-information is encountered before ipnum is exhausted, COPYL terminates with a return code of 8.

OSKIP=opnum * (all) or number of partitions (as specified by part or default) to be skipped on outfile before the copy operation is begun; default is 0. If a numeric value is specified and a higher-level partition delimiter or an end-of-information is encountered before opnum is exhausted, COPYL terminates with a return code of 8.

IRW=irw Inhibit rewind of input and output files. This applies to tape files only.

Y Files are not rewound prior to copying.

N Files are rewound prior to copying.

If IRW=irw is omitted, N is used.

NOCOMP Indicates that COPYL should not perform blank compression on the output file. If NOCOMP is omitted, COPYL performs blank compression.

Figure 4-11. COPYL Control Statement Format (Sheet 2 of 2)

DAYFILE - COPY THE JOB DAYFILE

The DAYFILE control statement copies the job dayfile. Depending on the option selected, it copies either the entire dayfile or only the portion of the dayfile written since the last DAYFILE statement was processed.

NOTE

The batch processor processes the DAYFILE statement. Therefore, the statement is available only in batch jobs. It is not a valid entry in an interactive session.

Specify on the DAYFILE statement the file to which the dayfile is copied. The file can be an attached permanent file or an existing temporary file. It cannot be a public file or a system pool file. If the file specified does not exist or is not attached, DAYFILE creates the file.

If a file is not specified on the statement, DAYFILE creates a file named OUTPUT and copies the dayfile to that file. After processing a statement, the batch processor renames the OUTPUT file so that it is part of the print family of files for the job (refer to Job Processing in chapter 3 of this manual). Therefore, if a file is not specified on the DAYFILE statement, the dayfile is printed with the job output.

DAYFILE copies until it encounters the end of the dayfile or the end of the file on which it is writing. After copying the dayfile, DAYFILE always reduces the file length to the length actually used. It reduces an existing file in the same way as it reduces a file it creates.

Figure 4-12 shows the DAYFILE control statement format.

DAYFILE, lfn/len, LO=option.

lfn Name of the file on which the dayfile is copied. It can be either an existing file or a new file created by the utility. If lfn is omitted, the file name is OUTPUT.

len Number of blocks to be allocated for the file. len can be designated by decimal or hexadecimal number representation. If in hexadecimal, the number must be preceded by the pound sign (#). If len is omitted and lfn is not an existing file, eight blocks are allocated.

LO=option Indicates the type of listing generated.

I Requests incremental copy: only that portion of your dayfile that is new since the last DAYFILE request is copied. If this is the first request, the entire dayfile is copied (same as F).

F Requests full copy: the entire dayfile is copied.

If LO=option is omitted, DAYFILE copies the entire dayfile.

Figure 4-12. DAYFILE Control Statement Format

DEFINE - DEFINE A PERMANENT FILE

The DEFINE statement defines a private permanent mass storage file.

Figure 4-13 shows the DEFINE control statement format. The first parameter must be the file name. File length, if specified, must be the second parameter. All other parameters are optional and can appear in any order.

If the file named on the DEFINE statement does not already exist as a local mass storage file, a new file is created as an attached permanent mass storage file. Each characteristic of the new file is given the default value unless a different value is specified using the corresponding optional DEFINE parameter.

If the file named on the DEFINE statement already exists as a local mass storage file, DEFINE changes the local file to a permanent file, but it does not change other file characteristics. All DEFINE parameters except the file name are ignored. To change file characteristics, use the SWITCH, PERMIT, or ROUTE utilities.

Upon successful completion of this operation, the message CREATED PERMANENT FILE or EXISTING LOCAL FILE MADE PERMANENT is sent to the job dayfile or the interactive terminal.

DEFINE, lfn/len, ACCESS=acs, RLMIN=rlmin, RLMAX=rlmax, NOEXTEND, NOSEGMENT, PACK=packid, PC=pc, RMD=rmd, RT=rt, SECURITY=lvl, SFO=org, TYPE=type, AU=blocks.

lfn Name of the mass storage file. lfn must be one through eight letters or digits, beginning with a letter (except for the name of a local drop file).

len Number of 512-word blocks initially allocated for the file (decimal or hexadecimal number between 1 and #FFFFFF).

ACCESS=acs Access permission set of the file owner (any combination of the following letters without separators).

- R Read permission
- W Write permission
- X Execute permission
- A Append permission
- M Modify permission

If ACCESS=acs is omitted, the default access permission set depends on whether the file is a new file or an existing local file. For a new file, DEFINE assumes ACCESS=RWXAM. For an existing file, DEFINE does not change its existing access permission set.

RLMIN=rlmin Minimum record length in bytes. If RT=F is specified, the minimum record length is ignored. If RLMIN=rlmin is omitted, the minimum length is one byte.

RLMAX=rlmax Maximum record length in bytes (fixed record length for F format records). If RLMAX=rlmax is omitted, the default maximum record length is an installation parameter value (released value, zero). A maximum record length of zero prevents writing on a direct access file.

Figure 4-13. DEFINE Control Statement Format (Sheet 1 of 3)

<u>NOEXTEND</u>	Indicates that the file cannot be extended. If NOEXTEND is omitted, the file can be extended as necessary. To create a file that will always remain contiguous, both the NOEXTEND and the NOSEGMENT options must be specified.
<u>NOSEGMENT</u>	Indicates that the initial file space allocated must be contiguous. If NOSEGMENT is omitted, the system can allocate initial file space in multiple segments. To create a file that will always remain contiguous, both the NOEXTEND and the NOSEGMENT options must be specified.
<u>PACK=packid</u>	Identifier of a pack in the device set on which the file is created. Pack identifiers are six characters long, left-justified, and blank filled. Excess characters are truncated. The pack parameter is ignored for existing local files. If PACK=packid is omitted, the system selects a pack.
<u>PC=pc</u>	ASCII padding character used to fill the working storage area. If PC=pc is omitted, the installation-defined default padding character (released value, blank) is used.
<u>RMD=rmd</u>	ASCII record mark character for R format records. If RMD=rmd is omitted, the installation-defined character [released value, ASCII US character (#1F)] is used.
<u>RT=rt</u>	Record format. If SFO=D is specified, the only valid record format is F. If RT=rt is omitted, the default format depends on the file organization. For sequential access files, the installation default format (released value, R) is used. For direct access files, F format is used.
	F ANSI fixed length
	R Record mark delimited
	U Undefined
	W Control word delimited
<u>SECURITY=lvl</u>	Security level (1 through 8). The specified security level cannot be greater than the security level of the job or interactive session. If SECURITY=lvl is omitted, the security level of the job or interactive session is used.
<u>SFO=org</u>	File organization. If SFO=org is omitted, the installation-defined default organization (released value, sequential access) is used.
	D Direct access
	S Sequential access

Figure 4-13. DEFINE Control Statement Format (Sheet 2 of 3)

<u>TYPE</u> =type	File type. If TYPE=type is omitted, the file is a physical data file.				
	<table> <tr> <td>C</td> <td>Controllee file</td> </tr> <tr> <td>P</td> <td>Physical data file</td> </tr> </table>	C	Controllee file	P	Physical data file
C	Controllee file				
P	Physical data file				
<u>AU</u> =blocks	Allocation unit. Allows the user to aid performance by giving the system a guideline on the integer number of 512-word blocks to allocate when the file is extended. The value range of blocks is 1 to 65,535. If the file is created and blocks is not a multiple of the DAU (Device Allocation Unit) for the device in which the first allocation occurs, blocks is rounded up to the next multiple of the DAU. If the file is already local, this parameter is ignored.				

Figure 4-13. DEFINE Control Statement Format (Sheet 3 of 3)

DEFINING A NEW FILE

If the file does not exist, DEFINE creates a new mass storage file. Using the length, disk pack residence, extendability, and segmentation specifications from the control statement, DEFINE allocates file space (refer to File Space Allocation in chapter 2 of this manual). The following lists the effect of each combination of the NOSEGMENT and NOEXTEND parameters.

<u>NOEXTEND</u>	<u>NOSEGMENT</u>	<u>Effect</u>
Omitted	Omitted	File has one or more segments. Noncontiguous segments can be added.
Specified	Omitted	File has one or more segments. It cannot be extended.
Omitted	Specified	File has one segment. Noncontiguous segments can be added.
Specified	Specified	File has one segment. It cannot be extended.

DEFINE also defines the following file attributes for a new file:

- Access permission set of the file owner
- File type (controllee or data)
- Security level
- Record format characteristics

DEFINE cannot define a file with a security level greater than the job or interactive session security level.

The retention period for the file is an installation option. The SWITCH control statement can be used to specify a particular number of days the file is to be retained on mass storage. The retention period determines the expiration date referenced by the DUMPF utility.

DIVERT - CHANGE THE DESTINATION OF AN OUTPUT FILE

The DIVERT statement allows you to change the destination LID of any file in the output queue whose original owner is the user number from which DIVERT is executing.

Output files are specified by their jdn or by the JN=jobname parameter. If jdn or jobname is not specified and if DIVERT is executed from within a batch job, the output of the batch job is changed to the LID specified by the ST parameter. If the new LID is associated with a different PID than the old LID, any routing information sent by the old host with the input batch file or specified by the JCS or I parameter of the MFQUEUE is ignored. In this case, the disposition of the output is dependent on the system defaults of the new host. For example, if MF1 and MF2 are LIDs associated with two different NOS systems and JOB1 is the output of a job submitted from MF1 but DIVERTed to MF2, the user number and family information sent with JOB1 is lost and it is likely that JOB1 will print on the default printer of MF2 using JOB1 as the banner.

DIVERT is executable from both batch jobs and interactive sessions. A successful DIVERT with either jdn or jobname specified will cause the appropriate output spooler application to start up (provided OUTPUT is ON and there are not too many applications running already).

Figure 4-13.1 shows the DIVERT control statement format.

DIVERT, { jdn JN=jobname } ,ST=newlid.	
<u>jdn</u>	Job descriptor number (1 through 2047). This parameter is optional and mutually exclusive with the JN=jobname parameter. The output-file-family with the specified jdn has its destination LID changed to newlid.
<u>JN=jobname</u>	Job name (or file name) as specified on the Q,0 command. This parameter is optional and mutually exclusive with the jdn parameter. The output-file-family(s) with job name specified by jobname has its destination LID changed to newlid.
<u>ST=newlid</u>	Logical ID (LID) of the remote host to which the output specified by jdn or jobname is to be sent. If neither jdn or jobname is specified, then the batch job output from which SUBMIT is executing is sent to newlid. This parameter is required.

Figure 4-13.1. DIVERT Control Statement Format

Upon successful completion, the following message is issued for each output-file-family that is diverted:

OUTPUT FILE jobname WITH JDN = jdn DIVERTED TO LID newlid

An unsuccessful completion of the DIVERT command will result in one of the following messages:

SYNTAX ERROR
SYSTEM MESSAGE #fc ERROR, RCODE + #rc, SSCODE = #ss
NO USER OWNED OUTPUT FILES FOUND WITH JDN = jdn
NO USER OWNED OUTPUT FILES FOUND WITH JN = jobname
newlid IS NOT A VALID DESTINATION LID
JDN OR JN MUST BE SPECIFIED WHEN INTERACTIVE
JDN AND JN PARAMETERS ARE MUTUALLY EXCLUSIVE

DMAP - PROVIDE INFORMATION ON LOCATION OF FILE SEGMENTS

DMAP provides information on the locations of segments for individual files. It is not intended to be a general file listing utility, but rather to be an aid in checking specific file residence.

If you are an individual nonprivileged user, DMAP is used to list the segments of your files that are resident on a particular pack (PN=parameter). If a device is to be removed from the system configuration, you can list the files that would be affected by this and move them elsewhere. If you know the device sets to use for the DS=parameter, all of the files on the set are listed.

If you are a privileged user, DMAP lists all file segments on the pack(s) or device set(s) specified. This listing can be sorted several ways to provide information on the overall pattern of disk usage, on the amount of fragmentation on files, or on the distribution of your files.

For allocated space, the file names, type, owner, PFI entry offset, date of origin, date of last access, highest byte written, length, device set name, pack name, and sectors allocated are listed in the output. DMAP traces the allocation pointers to other devices, if necessary, to obtain this information. If the information is not available because a device is down, it is indicated as an orphan segment, and the pack name that is unavailable is indicated in the listing. If the PN=parameter is selected and the file has segments on other devices that were not selected, an asterisk is appended to the pack name.

DMAP expects PFI_{nn} (where nn is the pack number) files to be public. If they aren't, DMAP will fail when it encounters the pack where the associated PFI_{nn} file is non-public. The PFI_{nn} file may have been made non-public for performance reasons. Check with your site's system's analyst if this occurs.

Figure 4-14 shows the DMAP control statement format.

DMAP, { PN=packname } { DSET=devset }	,LO=options,LIST=lfn
<u>PN</u> =packname	List of pack names for which disk space usage is to be listed. If the DS parameter is not specified, PN= is required. * specifies all packs.
<u>DSET</u> =devset	List of device sets for which disk space usage is to be listed. If the PN parameter is not specified, DS= is required. * specifies all device sets.
<u>LO</u> =options	options specifies the list of primary and secondary sort keys for the output. All entries are sorted on first (primary) keys; then the second key (if present) is used to sort and split the listing into smaller groupings. D The file information is printed as a function of position on disk. A The file information is printed for each file, and the files are listed alphabetically. U The file information is listed by user number. For each user number, the files are listed as for the A or D option. Default is LO=A. Permissible options are D, A, U, AD, AU, UD, and UA.
<u>LIST</u> =lfn	This parameter specifies the file to which the output listing is written. The default file name is OUTPUT.

Figure 4-14. DMAP Control Statement Format

```

-----
                PACK = PACK 37
                DAU VALUE =    4
                DFS INDEX = OCFC

(OCFC) = 0D3C371709180037
(OCFD) = 000100C800010129
(OCFE) = 0001029C000102A2
(OCFF) = 00010C5D00010CE1
-----

                PACK = PACK37
                DAU VALUE =    4
                DFS INDEX = OD3C

(OD3C) = 0D5837370CFC0037
(OD3D) = 00010DB800010DC9
(OD3E) = 00010E2300010FB6
(OD3F) = 00010FE400011544
-----

```

<u>Option</u>	<u>Description</u>
TY	The file type (physical data or virtual code)
PFIADR	The bit offset of the entry for the file into the PFI
DORG	The date of file creation
DOLA	The date of last access (the most recent attachment with write, append, or modify access)
HBW	The highest byte written to the FLEN is the allocated file length in 512-word blocks
DSET	The device set name
PN	The pack name

If a file was created prior to the system 2.2 release, an asterisk follows the name of the file.

If a file has been marked purge-only, two asterisks will immediately follow the file name. This file could have been created before or after the 2.2 release.

DROP - REMOVE A JOB FROM A QUEUE

The DROP statement allows you to drop user-owned executing jobs or queued files. (User-owned executing jobs have the same user number as that under which the DROP command is executing.) If the DC=q parameter on this control statement is not specified, only the input queue is searched. You cannot drop the job that is executing the DROP command nor can it drop an interactive session. You may use either jdn or JN=jobname to specify the job. If the jdn or jobname specified is in the input queue, the associated input file is evicted and an output file with the same jdn is created containing the following dayfile message:

JOB EVICTED FROM INPUT QUEUE BY USER.

If the jdn or jobname is in the execute queue, the job's currently executing task is interrupted and the following message is written to the job dayfile:

JOB DROPPED BY USER.

Reprieve is invoked if enabled and the job goes to EXIT processing. If the jdn or jobname is in the output queue, the associated output file family is evicted without notice. If KILL is specified and the job specified by the jdn or jobname is in the execute queue, the job is dropped without EXIT processing and the following message is written to the job dayfile:

JOB KILLED BY USER.

If more than one queue is specified, the queues are searched in the following order: input, execute, and output.

Figure 4-14.1 shows the format of the DROP control statement.

$\text{DROP, } \left\{ \begin{array}{l} \text{jdn} \\ \text{JN=jobname} \\ \text{JN=*} \end{array} \right\} , \text{KILL, DC=q.}$	
<u>jdn</u>	Job descriptor number (1 through 2047). This parameter is mutually exclusive with the JN parameter. The job specified by jdn is dropped from the queue(s) specified by q. Either the jdn or JN parameter must be specified.
<u>JN=jobname</u>	Job name (or file name). This parameter is mutually exclusive with the jdn parameter. The job(s) whose job name is jobname is dropped from the queue(s) specified by q.
<u>JN=*</u>	Drop all user-owned queue files in the queue(s) specified by q. This parameter is mutually exclusive with the jdn parameter.
<u>KILL</u>	This parameter is optional and applies only when DC=E. If specified, the job specified by jdn or jobname is terminated without EXIT card processing (that is, the job is killed).
<u>DC=q</u>	Specifies the queue(s) where the job(s) specified by jdn, jobname, or * resides. This parameter is optional. If q is not specified, only the input queue is searched. Multiple queues are searched by specifying combinations of I, E, and O or by specifying *, where: I Input queue E Execute queue O Output queue * All queues are searched

Figure 4-14.1. DROP Control Statement Format

If the DROP statement executes successfully, the following message is issued for each job that is dropped:

```
JOB jobname WITH JDN = jdn WAS DROPPED FROM queue QUEUE
```

If the DROP statement did not complete successfully, one of the following error messages is issued:

```
SYNTAX ERROR  
SYSTEM MESSAGE #fc ERROR, RCODE = #rc, SSCOPE = #ss  
NO USER OWNED JOB FOUND WITH JDN = jdn  
NO USER OWNED JOB FOUND WITH JN = jobname  
CAN NOT DROP INTERACTIVE JOB  
JOB CAN NOT DROP ITSELF
```

DUMPF - ARCHIVE FILES

The DUMPF control statement archives permanent files or queue files. File archiving is the process of copying permanent files to backup storage and reloading the backup copies if needed. File archiving preserves a backup copy in the event that the original copy is inadvertently destroyed.

A nonprivileged user can archive only attached private or pool files. A privileged user can archive all permanent files stored on the CYBER 200 system except attached private files and files belonging to user numbers 1 through 15. A system user can list information about the I/O queues. Neither privileged or nonprivileged users can archive the system user directory or files with the following reserved names:

```
JOBFILE
Q5DAYFLE
Q6DLFEOT
Q5JOBFILE
Q5JRTHRF
Q6DLFTRC
Q5SDFLFN
Q6OUTPUT
*AF
*AF2
*HISTORY*
```

Files with reserved file names can be archived by switching or copying them to a nonreserved file name and then archiving them. Similarly, files under user numbers 1 through 15 can be archived by switching or copying the files to a different user number.

NOTE

The access directory for a file is saved only if the user archiving the file is privileged. The access directory is not saved when a nonprivileged user archives the file. Therefore, when a file archived by a nonprivileged user is reloaded, its access permissions must be redefined.

DUMPF executed in update mode (SELECT=U) should be used to dump to tape and not to mass storage. If dumping to mass storage during production hours, the system table FILEI may get full and impact the operation of the system.

The DUMPF control statement bypasses dumping permanent files if either of the following situations exists:

- Any user has the permanent file attached with write, modify, or append access permission set.
- Any user has a pool attached and the pool file is currently open with write, modify, or append access permission set.

DUMPF can execute concurrently with other tasks, including other DUMPF tasks.

DUMPF executed with the DEVICE=0 parameter allows privileged and nonprivileged users to purge files without dumping the permanent file to a device. Using the DATE, SELECT, and TIME parameters along with DEVICE=0 allows users to purge files that have or have not been modified, accessed, created, or expired within a specified date and/or time period.

The DUMPF control statement format is shown in figure 4-15. All parameters except the first can appear in any order. The first parameter, if specified, must be a list of file names.

The first format shown in figure 4-15 is used when the RHF application, DLF, dumps the files to a remote system. The second format is used when the files are archived on CYBER 200 mass storage or tapes.

NOTE

If a production file is dumped by any user other than the site security administrator (refer to chapter 7 of the Installation Handbook), the file will not retain its production status when reloaded.

Format for Front-End File Archiving

DUMPF,lfn-list,USER=userno,POOL=plist,DSET=devset,PACK=packlist,ACCOUNT=alist,
JCAT=jcatlist,LID=lidlist,SELECT=opts,DATE=mmddy,TIME=hhmm,LO=x,LIST=lfn/len,
ST=stid,SI=setid, { JCS=strings }
 { INPUT=lfn } .

Format for CYBER 200 File Archiving

DUMPF,lfn-list,USER=userno,POOL=plist,DSET=devset,PACK=packlist,ACCOUNT=alist,
JCAT=jcatlist,LID=lidlist,SELECT=opts,DATE=mmddy,TIME=hhmm,VERIFY=pt,LO=x,
LIST=lfn/len,DEVICE=device,VSN=id-list,TF=tf,DENSITY=den,RP=days,IU=iu.

File Specification Parameters

lfn-list List of 1 through 128 file names separated by commas. The specified files are assumed to belong to any or all user numbers specified by the USER parameter and/or any or all pools specified by the POOL parameter. If omitted, all files belonging to userno and/or plist are archived. If SEL=0 is specified, lfn-list identifies the last-group-files(s) of the output-file-family(s) to be archived. If SEL=0 is specified and lfn-list is omitted, DUMPF archives all output-file-families.

USER=userno Private file owners.

For a nonprivileged user:

 userno User number of the nonprivileged user.

For a privileged user:

 u-list List of 1 through 128 user numbers separated by commas.

 * All file owners, private, pool, and public.

For a system user who has specified SEL=I or SEL=0:

 u-list List of 1 through 128 user number(s) that queued the files (original owner). If omitted, DUMPF will archive the queue files of all user numbers.

If USER=userno is omitted and the POOL parameter is not specified, DUMPF archives files belonging to the user number under which DUMPF was run.

POOL=plist List of 1 through 128 pool names separated by commas.

DSET=devset Allows files to be dumped to a specific device set. devset is a list of 1 through 128 device sets (DVSTnn) separated by commas. If more than one device set is specified, the device sets will be used in the order they appear in the parameter list, with the first being filled before the next one is used.

Figure 4-15. DUMPF Control Statement Format (Sheet 1 of 5)

File Specification Parameters

- PACK=packlist Allows files to be dumped on a pack basis. packlist is a list of 1 through 128 pack names (PACKnn) separated by commas. Only those files beginning on a specified pack are archived. Those that continue from another pack are not archived.
- ACCOUNT=alist For a nonprivileged user, alist is a list of one to seven account identifiers separated by commas. You must be validated for all specified account identifiers in order to archive files with these accounts. For a privileged user, alist is a list of 1 through 128 account identifiers separated by commas. Only files with the specified accounts are archived.
- JCAT=jcatlist List of 1 through 64 job categories separated by commas. This parameter is allowed only if SEL=I or SEL=O is specified and applies only to the input queue. If this parameter is omitted, files belonging to all job categories in the input queue are archived.
- LID=lidlist List of 1 through 128 destination LIDs for input or output files. This parameter is allowed only if SEL=I or SEL=O is specified. If this parameter is omitted, all queue files are archived regardless of their destination LIDs.
- SELECT=opts File characteristics of all files dumped (any combination of the following letters without separators). A file must meet all characteristics specified in order to be dumped.
- A Files accessed on or after the date and time specified by the DATE and TIME parameters. An access is defined as an open.
 - C Files created on or after the date and time specified by the DATE and TIME parameters.
 - I Files in the input queue. Only the system user is allowed to select this option. The I option is mutually exclusive with the PO parameter.
 - M Files modified on or after the date and time specified by the DATE and TIME parameters.
 - N Reverses the meaning of the A, C, or M options. For example, NC specifies files not created since the date and time specified. This option may appear anywhere in the string but always reverses the meaning of all characters specified.

Figure 4-15. DUMPF Control Statement Format (Sheet 2 of 5)

File Specification Parameters

- O Files in the output queue. Only the system user is allowed to select this option. The O option is mutually exclusive with the PO parameter.
- X Files expired. A file expires when more days have passed since its creation date than the number of days in the retention period for the file.
- P Purge the file after successfully dumping it. If its dump is not successful, the file is not purged. If the DEV=0 parameter is specified, files are purged and not dumped. Only the pool boss or a privileged user can purge a pool file.
- U Indicates update mode. Only files created or modified since the last update dump are selected. This option is mutually exclusive with other SELECT options or date and time parameters. Only a privileged user can use this option.

If SELECT=options is omitted, DUMPF assumes no options.

DATE=mmddy

Date used by the A, C, and M options on the SELECT=options parameter. The first two digits of the date indicate the month, the next two digits the day of the month, and the last two digits the last two digits of the year.

If DATE=mmddy is omitted, DUMPF uses the current date.

TIME=hhmm

Time used by the A, C, and M options on the SELECT=options parameter. hh is the hour, based on a 24-hour clock. mm is the minute in the hour.

If TIME=hhmm is omitted, DUMPF uses midnight.

Verification Parameters

VERIFY=opt

Verify the integrity of the archival medium just written. Mis-compare errors are noted in the dayfile and verification is halted after a threshold of errors is reached. If any error and if SEL=P was selected, no files will be purged by DUMPF. The extent of the verification is determined by the opt value as follows:

- Q Quick verification. The archival medium is scanned, selected fields are checked for consistency, and the archival medium contents are compared with the list of files DUMPF dumped.
- F Full verification. The quick verification is performed plus the data of each file is read, to ensure it is readable. The length of each file read is compared with the length of the file dumped. However, a bit for bit comparison of files is not performed. This option may take 2 - 30 times as long (or longer) as the Q option depending on the size of the files and the archival medium involved.

If this parameter is not specified, no verification is performed.

Figure 4-15. DUMPF Control Statement Format (Sheet 3 of 5)

Listing Parameters

LO=x Audit information required.

 F Full audit.

 P Partial audit.

 If LO=x is omitted, DUMPF writes partial audit information.

LIST=lfm/len Listing file specifications.

 lfm File name (one to eight letters or digits, beginning with a letter). If lfm is omitted, DUMPF uses file OUTPUT.

 len File length in 512-word blocks. If len is omitted, the file length is #40 blocks.

For Front-End File Archiving Only

ST=stid RHF logical identifier of the other system (three ASCII characters). This parameter is required.

SI=setid Set identifier of the archive storage on the other system. This parameter is required. It must be a name of one to six letters or digits beginning with a letter.

 On NOS/BE, the SI parameter is the multifile set name and must be the same name as specified on the VSN directive. For the IBM remote host, the SI parameter is ignored.

JCS=strings List of one to ten text strings sent to the other system. Each string must be delimited by double quote (") characters. Strings within the list are separated by commas.

 You cannot use this parameter if you use the INPUT parameter. If both JCS and INPUT are omitted, then I=INPUT is assumed.

INPUT=lfm Name of the CYBER 200 file containing the text strings to be sent to the other system. A text string in the file must appear as it would if entered on the JCS parameter without the " delimiters.

 You cannot use this parameter if you use the JCS parameter. If both JCS and INPUT are omitted, then I=INPUT is assumed.

For CYBER 200 File Archiving Only

NOTE

The following parameters are ignored if specified with RHF file archiving parameters.

Figure 4-15. DUMPF Control Statement Format (Sheet 4 of 5)

<u>DEVICE=</u> device	Device type used to store archived files.
	MS Mass storage.
	NT Magnetic tape.
	0 No device to assign (mutually inclusive with SELECT=P).
	If DEVICE=device is omitted, the installation defined default type (released value, MS) is used. When DEVICE=0 is supplied, the files that are selected are purged without being archived. The VSN parameter cannot be specified if DEV=0.
<u>VSN=</u> id-list	Archive storage device identifier (a list of one through six device sets in the format DVSTnn if DEVICE=MS, or a list of 1 to 255 tape volume VSNs if DEVICE=NT). VSN is a required parameter if DEVICE is MS or NT. If DEVICE=0, the VSN parameter is not allowed.
	If the listed devices are not sufficient, the operator must assign additional devices.
<u>For CYBER 200 File Archiving Only</u>	
<u>TF=</u> tf	Tape format (for tapes only):
	V Variable block size format with block size set to 8K.
	LB Large block size format.
	If TF=tf is omitted, DUMPF defaults to TF=V. The LB format is the format used by pre-2.3 DUMPF. This parameter is mutually exclusive with DEV=MS.
<u>DENSITY=</u> den	Recording density (for tapes only):
	PE 1600 cpi.
	GE 6250 cpi.
	If DENSITY=den is omitted and DEVICE=NT is specified, the installation-defined default density (released value, 6250 cpi) is used. If the density specified or the default density does not match the density on the tape, processing continues with the density specified on the tape.
<u>RP=</u> days	Retention period in days (1 through 999).
	If RP=days is omitted, the default set by an installation parameter is used.
<u>IU=</u> iu	Inhibit unload option indicating whether the system unloads a tape volume when the utility is complete. This applies to tape files only.
	<u>Y</u> Does not unload tape volume.
	<u>N</u> Unloads tape volume.
	If IU=iu is omitted, N is used.

Figure 4-15. DUMPF Control Statement Format (Sheet 5 of 5)

VSN=id-list Archive storage device identifier (a list of one through six device sets in the format DVSTnn if DEVICE=MS, or a list of 1 to 255 tape volume VSNs if DEVICE=NT). VSN is a required parameter if DEVICE is MS or NT. If DEVICE=0, the VSN parameter is not allowed.

If the listed devices are not sufficient, the operator must assign additional devices.

For CYBER 200 File Archiving Only

TF=tf Tape format (for tapes only):

V Variable block size format with block size set to 8K.

LB Large block size format.

If TF=tf is omitted, DUMPF defaults to TF=V. The LB format is the format used by pre-2.3 DUMPF. This parameter is mutually exclusive with DEV=MS.

DENSITY=den Recording density (for tapes only):

PE 1600 cpi.

GE 6250 cpi.

If DENSITY=den is omitted and DEVICE=NT is specified, the installation-defined default density (released value, 6250 cpi) is used. If the density specified or the default density does not match the density on the tape, processing continues with the density specified on the tape.

RP=days Retention period in days (1 through 999).

If RP=days is omitted, the default set by an installation parameter is used.

Figure 4-15. DUMPF Control Statement Format (Sheet 5 of 5)

SPECIFICATION OF FILES TO BE ARCHIVED

The set of files that DUMPF archives can be specified by name or by attributes. The set of files must have all the attributes specified.

The USER and POOL parameters specify file ownership, the DSET and PN parameters can specify file residence, and the SELECT, DATE, and TIME parameters can specify file usage and age.

A maximum of 256 private files and 256 files per pool can be dumped.

NOTE

No more than 2048 private files and/or pool files can be dumped at one time. If you need to dump more than this number, do two or more dumps, using a different file specification for each. For example, specify a different device set for each dump, or dump users and pools separately.

If no file names are specified and the USER, POOL, ACCOUN, and SEL=I or SEL=O parameters are omitted, DUMPF archives files belonging to the user number under which DUMPF is executed (provided that SEL=I or SEL=O is specified).

Table 4-2 in this chapter summarizes the interaction of the USER and POOL parameters.

If a file cannot be archived, DUMPF returns an appropriate message and continues processing with the next file.

Files archived are listed in the user dayfile along with the user number or pool to which the files belong. If two or more consecutive files belong to a single user number or pool, only the first file lists the user number or pool.



ARCHIVE FILE FORMAT

The same archive file format is used whether the file is stored on a remote system or on the CYBER 200 system.

The first block of each archived file contains the data needed to reload the file. The archived file format is shown in figure 4-16.

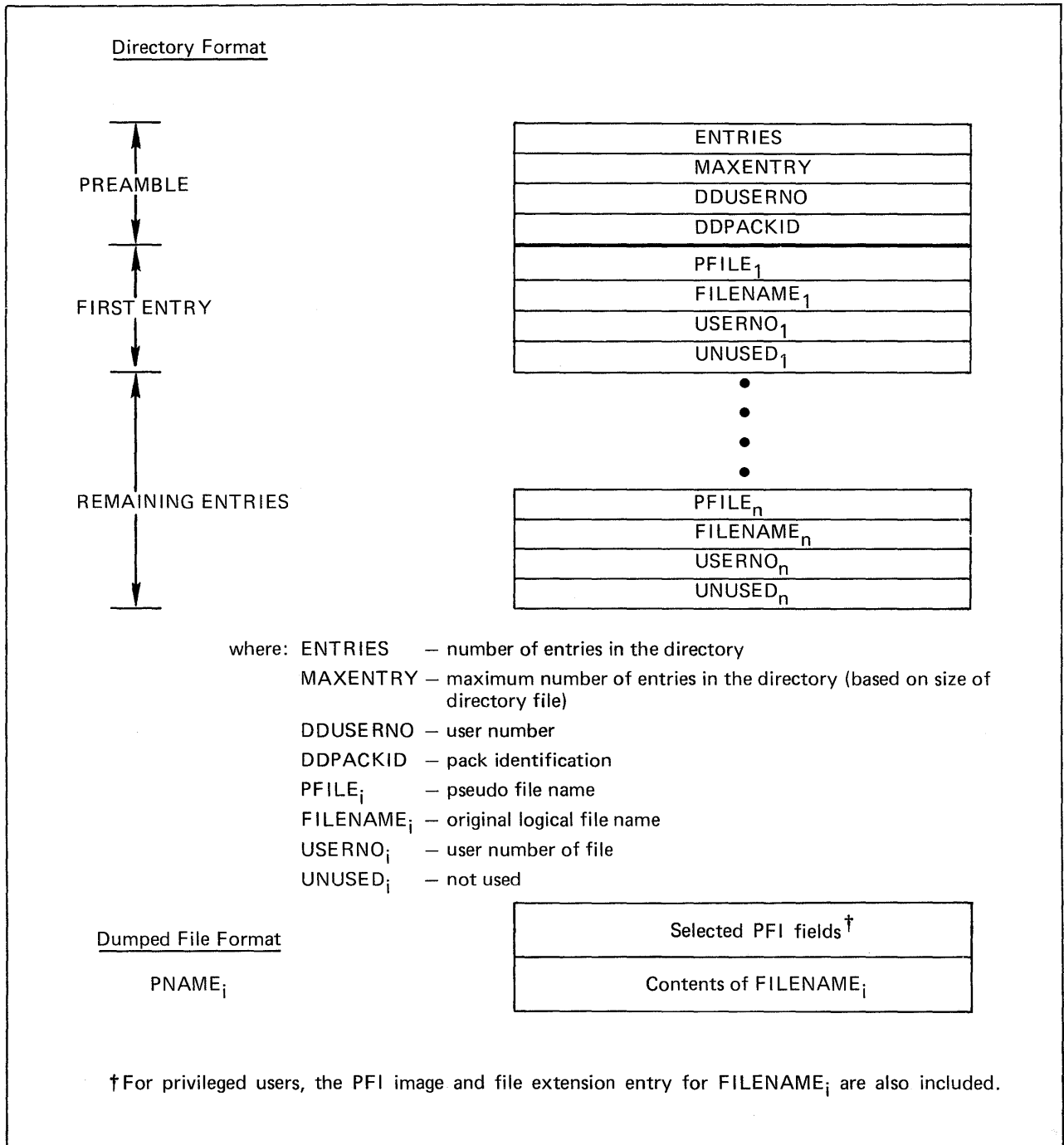


Figure 4-16. Directory/Dumped File Format

The information in the first block of each archived file includes the contents of selected fields in the permanent file index entry for the file.

If you are a privileged user and you archive the file, the first block also contains the following information:

- A copy of the unformatted permanent file index (PFI) entry as it exists after DUMPF opens the file
- A copy of the permanent file index extension entry if an access directory exists for the file

The access fields in the permanent file index entry are updated when DUMPF opens the file. Therefore, the access field information differs, depending on whether you are or are not privileged. This means that if you specify the A option, the last access date and time used differ, depending on whether you are or are not privileged.

ARCHIVING TO A FRONT-END SYSTEM

If the RHF application program is present on the system, it interprets the ST, SI, JCS, and INPUT parameter specifications on the DUMPF control statement. DUMPF uses the ST parameter specification to determine the remote system on which the file copies are stored. The SI, JCS, and INPUT parameter specifications determine how the file copies are stored on the remote system.

DUMPF passes a text string to the remote system before it sends the file copies to be stored. The string is specified on the JCS parameter or in the file specified on the INPUT parameter. The required content of the text string depends on the RHF software in the remote system. For more information, refer to the RHF documentation for the remote system.

ARCHIVING TO CYBER 200 MASS STORAGE

When archiving to CYBER 200 mass storage, use the DEVICE and VSN parameters to specify the disk packs on which the file copies are stored.

The VSN parameter lists the device set identifiers of the device sets to be used. The device sets are used in the order listed on the parameter. DUMPF sends a message to the job dayfile or to the interactive terminal when it switches device sets. If DUMPF exceeds the specified sets, the operator is prompted for additional device set names.

When DUMPF archives files on CYBER 200 mass storage, the archived file copies are unattached private files. DUMPF maintains a directory file on each device set for the files dumped to that set. The name of the directory file is DVSTnn00 where nn is the archive storage device identifier (device set number). The directory contains the file names of all archived files for this user on the device set. The directory format is shown in figure 4-16.

If, during DUMPF processing, a directory entry already exists that has the same name and owner as a file to be archived, the existing archived file with that name and owner is destroyed. DUMPF copies the file to be archived and creates a new directory entry.

The name of each archived file (called a pseudo file) is specified as DVSTmmnn. mm is the archive storage device identifier. nn is the file sequence number in hexadecimal representation. If the sequence number exceeds #FF, the sequence number begins overwriting the first part of the name (for example, file number #2F3 would be archived on DVST13 as DVST12F3).

Using the DEVICE=0 parameter, both privileged and nonprivileged users may purge selected files without having to dump the permanent files to a device first.

ARCHIVING TO CYBER 200 ON-LINE TAPES

To archive files on CYBER 200 on-line tapes, specify the DEVICE=NT parameter on the DUMPF statement. Also, specify the VSNs of the tape volumes on the VSN parameter. Optionally, you may specify the recording density on the DENSITY parameter and the retention period on the RP parameter. If the density of the tape does not match that specified by the density parameter, DUMPF uses the density of the tape and issues an appropriate error message.

The tape volumes specified on the VSN parameter are used in the order listed on the parameter. DUMPF sends a message to the job dayfile or to the interactive terminal when it switches tape volumes.

DUMPF writes the archived files as a multifile set. It generates a unique 17-character file identifier for each file in the set. Each file identifier generated has the following format:

yfilenameusername	
y	File owner (U for private file, P for pool file)
filename	File name, right-justified with zero character fill
username	ASCII user number or pool name, right-justified with zero character fill

For example, if a private file named MYFILE and belonging to user 012306 is archived, its file identifier is U00MYFILE00012306.

For the 2.3 release, the default tape format used by DUMPF was changed from LB to V with a block size of 8K. This was done to enhance reliability for both writing and reading on-line tapes. To maintain compatibility with pre-2.3 releases, a tape format parameter has been added to both DUMPF and LOADPF. Under normal use, the default format of V should be used. However, if you plan to create a tape that must be accessible to a pre-2.3 system, specify TF=LB.

DUMPF OUTPUT

The LO parameter on the DUMPF control statement determines whether DUMPF produces a full or a partial output listing. A full listing produces all of the headings described next, while a partial listing contains only the first 13 headings. A full listing does not exceed 132 characters, excluding the carriage return, and a partial listing does not exceed 80 characters, excluding the carriage return. Dates appear as month, day, and year. Time appears in a 24-hour format. All values are decimal unless noted otherwise.

The following are the column headings used in a full DUMPF listing and the information given under each heading.

<u>Heading</u>	<u>Description</u>
VSN	Volume serial number: this field is printed only the first time a file is dumped to a VSN or when the report goes to a new page.
FSN	File sequence number: hexadecimal count of files dumped.
NAME	File name.
OWNER	File owner: individual user number, public user number (0), or pool name. If SEL=I or SEL=O is specified, then the user number of the original file owner is listed.
TYP	File type: virtual code (VC) or physical data (PD).
FC	File category: batch input file (B), input queue file (I), output queue file (O), user file (U), system-generated drop file (S), or not defined (N).
RT	Record type: ANSI fixed length (F), record mark delimited (R), undefined (U), control word (W), system block (B), or lower CYBER (L).
BT	Blocking type: character count (C), internal (I), or record count (K).
ACS	Access permission set: read (R), write (W), execute (X), append (A) and/or modify (M) permissions, no permissions (NONE), or purge-only (PURGE). DUMPF lists the owner's access permission set for private files and the general access permission set for pool and public files.
EXT	File allocation: segmentable (S) and/or extendable (X).
SL	Security level: 1 through 8.
DEVICE	Device name of mass storage file. An asterisk following the device name will indicate that a portion of the file resides on another disk.
DSET	Name of device set.
FLEN	Number of 512-word blocks in file.
FACT	Accounting information.
DORG	Creation date (date of origin).
TORG	Creation time (time of origin).
DOLA	Date of last file access.
TLR	Time of last file access.
DOLM	Date of last file modification.
TOLM	Time of last file modification.
EXP	Expiration date (creation date plus retention period).

If SEL=I or SEL=O is specified, the TYP column is deleted and the following column headings replace ACS and EXT:

<u>Heading</u>	<u>Description</u>
LID	Destination LID for output and/or input queue files.
JCAT	Job category of input queue files. For all other file types, this field is left blank.

Figure 4-17 shows an example of an DUMPF output listing as produced by the following control statement:

```
DUMPF,U=*,AC=ACCTN01,ACCTN02,ACCTN03,DEV=NT,VS=CY2091,CY2088,LO=F.
```


CYBER 200 DUMPF DMP2219 -USER 14000 06/17/86 13.29.21																					
VSN	FSN	NAME	OWNER	TYP	FC	RT	BT	ACS	EXT	SL	DEVICE	DSET	FLEN	FACT	DORG	TORG	DOLA	TLR	DOLM	TOLM	EXP
CY2091	1	TRACE	10955	PD	U	R	C	XMARW	X	1	PACK3B	DVST3B	16	ACCTN03	028685	931	050885	948	020885	932	031085
	2	CF639B	10955	PD	U	R	C	XMARW	X	1	PACK3B	DVST3B	80	ACCTN03	020685	927	020685	931	020685	927	030885
	3	FT70249B	FTNDROP	PD	U	R	C	XMARW	X	1	PACK3B	DVST3B	80	ACCTN03	102284	1357	112184	1321	102284	1357	112184
	4	MAILIST	09151	PD	U	R	C	RW	X	1	PACK3B	DVST3B	16	ACCTN03	032985	927	060785	1014	060785	1007	042885
	5	V3	09151	PD	U	R	C	XMARW	X	1	PACK3B	DVST3B	16	ACCTN03	072484	1315	052985	846	052985	846	082384
	6	RENAME	BOBPOOL	VC	U	U	C	XR	X	1	PACK3B	DVST3B	50	ACCTN03	061385	1250	061385	1251	061385	1250	071385
	7	DELSRC	10011	PD	U	R	C	RW	X	1	PACK3B	DVST3B	16	ACCTN03	052485	1451	052985	1046	052885	923	062385
	8	PFDL	BOBPOOL	VC	U	U	C	XR	X	1	PACK3B	DVST3B	82	ACCTN03	051585	1503	061685	2356	051585	1503	061485
CY2088	9	DIAG22	9151	PD	U	R	C	XMARW	X	1	PACK3B	DVST3B	16	ACCTN03	031185	1031	061785	812	061285	833	041085
	A	C2700L	14000	PD	U	R	C	XMARW	X	1	PACK3B	DVST3B	16	ACCTN01	053185	618	053185	618	053185	618	063085
	B	C2700TS	14000	VC	U	U	C	XMARW	SX	1	PACK3B	DVST3B	320	ACCTN01	053185	618	053185	618	053185	618	063085
	C	CEBIN	10955	PD	U	R	C	XMARW	X	1	PACK3B	DVST3B	32	ACCTN03	041285	943	050985	1002	041285	943	051285
	D	MINK	POOLVRF	VC	U	U	C	XMARW	X	1	PACK3B	DVST3B	368	ACCTN03	041285	947	041585	933	041285	947	051285
	E	CG520L	14000	PD	U	R	C	XMARW	X	1	PACK3B	DVST3B	16	ACCTN01	053185	624	053185	624	053185	624	063085
	F	CG520TS	14000	VC	U	U	C	XMARW	X	1	PACK3B	DVST3B	320	ACCTN01	053185	624	053185	624	053185	624	063085
	10	CONNECT	BP22	VC	U	U	C	XR	X	1	PACK3B	DVST3B	64	ACCTN03	061085	1809	061085	1810	061085	1809	071085
	11	DISCONT	BP22	VC	U	U	C	XR	X	1	PACK3B	DVST3B	64	ACCTN03	061085	1809	061085	1811	061085	1809	071085
	12	DNAD	9151	VC	U	U	C	XR	X	1	PACK3B	DVST3B	128	ACCTN03	061185	1452	061185	1456	061185	1452	071185

Figure 4-17. DUMPF Output Example

EDITPUB - ADD OR DESTROY PUBLIC FILE

The EDITPUB control statement is valid only for privileged user numbers. It adds or destroys a public file.

The files specified on the N parameter (the files to become public files) must be attached private or pool files. Only the pool boss can issue an EDITPUB statement for a pool file.

The EDITPUB control statement format is shown in figure 4-18.

EDITPUB,	$\left\{ \begin{array}{l} \text{D=lfm-list} \\ \text{L} \end{array} \right\}$,N=lfm-list,P=lfm-list,ACCESS=acs,VRI=index.
<u>D</u> =lfm-list		List of public files to be destroyed (1 through 16 names, separated by commas).
<u>L</u>		Files to be destroyed are specified interactively. The L parameter can be specified at an interactive terminal only.
<u>N</u> =lfm-list		List of files to be added to the public file list without privileged status (1 through 16 names, separated by commas).
<u>P</u> =lfm-list		List of files to be added to the public file list with privileged status (1 through 16 names, separated by commas).
<u>ACCESS</u> =acs		Access permission set for each file specified by the N and P parameters (any combination of the following letters without separators).
	R	Read permission
	W	Write permission
	X	Execute permission
	A	Append permission
	M	Modify permission
		If ACCESS=acs is omitted, EDITPUB assumes ACCESS=RX.
<u>VRI</u> =index		Index into the Variable Rate Table in the range 1 through 255, for public files being added with the call. If VRI=index is omitted, the system uses an index of 0.

Figure 4-18. EDITPUB Control Statement Format

EDITPUB cannot destroy a file open to a task. If it cannot destroy a file, it returns an error message to the job dayfile or interactive terminal.

If a file that is being made public has the same name as an existing public file (that is, if a file is being replaced), an EDITPUB statement can both destroy the existing public file and add the new public file. For example, to replace public files X and Y, the following control statement is appropriate:

```
EDITPUB,D=X,Y,N=X,Y.
```

When the utility is called with the L parameter from an interactive terminal, it displays the name of each public file in turn and waits for one of the following terminal user responses:

<u>User Enters</u>	<u>Result</u>
D	Destroy file
Carriage return	Retain file
STOP	Terminate utility

VARIABLE RATE INDEX SPECIFICATION

If the variable rate index (VRI) parameter is used, at least one of the N or P parameters must be used. In this case, all files being made public in this control statement must be controllees, and the VRI parameter applies to all. Files made public using the VRI parameter do not retain read or write access.

If both the L and VRI parameters are used from an interactive terminal and the user response indicates that any file is to be retained, the VRI for that file is not reset to the VRI parameter value. The VRI file index entry for any file is 0 until modified by a VRI specification on an EDITPUB statement.

EXIT - SET ABNORMAL TERMINATION PATH

The EXIT control statement establishes the point at which the batch processor continues job processing after a task returns an abnormal termination code.

The EXIT control statement is valid only in a batch job. It is executed directly by the batch processor.

The EXIT control statement format is shown in figure 4-19. More than one EXIT control statement can appear in a job.

```
EXIT.
```

Figure 4-19. EXIT Control Statement Format

When abnormal job termination is initiated, (refer to Abnormal Job Termination in chapter 3 of this manual), the batch processor searches subsequent statements and continues statement processing with the control statement following the first EXIT or PROCEED encountered. If no EXIT or PROCEED control statement exists, job processing ends.

If the EXIT statement is encountered during normal job advancement to the next control statement, job processing ends normally at the EXIT statement.

The threshold value is set to 255 when an EXIT or PROCEED control statement establishes the execution path.

If control transfers to the path established by an EXIT or PROCEED control statement because the job time limit is reached, no time is available for user job tasks after the EXIT or PROCEED statement. A short amount of time is available to the job for use by the batch processor.

FILES - LIST FILE INFORMATION

The FILES control statement lists information about files.

The PRIVATE, POOL, and PUBLIC parameters indicate the ownership category of the listed files. The USER parameter specifies the owner of the private permanent files listed.

The names of local and attached private files can be specified either after the FILES verb or on the PRIVATE parameter. The names of unattached private files must be specified on the PRIVATE parameter.

The following are examples of FILES statements.

Example 1:

The following statement lists information about all local and private files attached to a job, including files attached but not owned by you.

```
FILES.
```

Example 2:

The following statement lists information about all public files, all private files owned by you (including local, attached, and unattached files), and all pool files belonging to the attached pool MYPOOL.

```
FILES,PUBLIC=*,PRIVATE=*,POOL=MYPOOL.
```

Example 3:

The following statement lists information about the private permanent file named HERFILE, owned by user number 012306.

```
FILES,HERFILE,USER=012306.
```

Example 4:

Either of the following statements lists information about all files belonging to user number 012306 that you can access.

```
FILES,USER=012306.
```

```
FILES,PRIV=*,USER=012306.
```

The FILES control statement format is shown in figure 4-20. All parameters are optional and, with the exception of the initial file name list, can appear in any order.

FILES, lfn1-list, PRIVATE= { lfn2-list } *, PUBLIC= { lfn3-list } *, POOL=pool, lfn4-list,
USER=userno, LIST=outlfn.

lfn1-list List of local and attached private files (1 to 255 file names separated by commas).

PRIVATE= { lfn2-list } * Private files (local, attached, or unattached).

lfn2-list List of 1 to 255 file names separated by commas.

* All private files attached to the user's jobs.

PUBLIC= { lfn3-list } * Public files.

lfn3-list List of 1 to 255 file names separated by commas.

* All public files.

POOL=pool,
lfn4-list Pool and list of files belonging to the pool (pool name followed by 1 to 255 file names separated by commas). A pool name specified without a list of files specifies all files belonging to the pool. More than one POOL parameter can be specified on the statement.

USER=userno Owner of the specified private permanent files (six-digit user number). The caller must have access to the files.

This parameter is provided to list files owned by another user to which the caller has access. If the caller's user number is specified and private permanent files are to be listed, only that user's private permanent files with general access is listed.

If USER=userno is omitted and private files are to be listed, the FILES output includes private files, regardless of access permissions.

LIST=outlfn Optional name of the file on which output is written.

If LIST=outlfn is omitted, the default output file depends on whether FILES is executed within a batch job or as an interactive task. Batch job output is written on file OUTPUT; interactive task output is displayed at the terminal.

Figure 4-20. FILES Control Statement Format

FILES OUTPUT

FILES lists information for the files it finds. It then lists the names of the specified files it did not find, files to which you do not have access, or those whose security level is greater than the security level of the job or interactive session. These files are listed under the following heading:

FILES NOT FOUND

File information for the files found is listed in alphabetical order by file name. File information for files with the same name is listed in the following ownership category order: private local, private permanent, pool, and public.

If you attach the same permanent file in more than one job, all information columns are listed for the first job only. The access permission (ACS) and the job descriptor number (JDN) are the only columns listed for your subsequent attaches of that file. For example, see the file listing LVIRT22 in figure 4-21.

NAME	DUP	ACS	LEN	JDN	ORI.DATE	OWNER	TYPE	DT	FC	BT	RT	FO
2FILES		XRW	0000256	471	09/12/84	*LOCAL	VC	MS	S	C	U	S
ACWI*		XMARW	0000004	1465	09/12/84	*PERM	PD	MS	B	C	R	S
C1955		XMARW	0000001		07/25/84	*PERM	PD	MS	U	C	R	S
CYBIL		XMARW	0000994		04/28/84	*PERM	VC	MS	U	C	U	S
CYBSLIB		XMARW	0000800		04/28/84	*PERM	PD	MS	U	C	R	S
LVIRT22		XR	0000208	1465	08/23/84	*PERM	PD	MS	U	C	R	S
LVIRT22		XR		471								
Q5RHFTRC		XMARW	0000008		04/26/84	*PERM	PD	MS	U	C	R	S
RHFREF		XMARW	0001968		03/23/84	*PERM	PD	MS	U	C	R	S
SOSUP		XMARW	0000008		07/26/84	*PERM	PD	MS	U	C	R	S
SPSUP		XMARW	0000004		07/26/84	*PERM	PD	MS	U	C	R	S
TOOLUP		XMARW	0000004		08/09/84	*PERM	PD	MS	U	C	R	S

Figure 4-21. FILES Sample Output

If the USER parameter is specified, file information for files owned by another user is listed under the following heading:

FILES OF ALTERNATIVE USER

File information is listed under the following column headings:

<u>Heading</u>	<u>Description</u>
NAME	File name. An asterisk is appended to the file name if it is a production file.
DUP	Duplicate file name flag. An asterisk in this column indicates that at least one other file exists with the same name and owner.
ACS	Access permission set of the user: read (R), write (W), append (A), modify (M) and/or execute (X) permissions, no permissions (NONE), or purge-only (PURGE).

<u>Heading</u>	<u>Description</u>
LEN	Actual length of each segment of a mass storage file (decimal number of 512-word blocks). Partial files (when not all of the files are available) are indicated by a minus sign to the right of the LEN field.
JDN	Job descriptor number of a blank field. It indicates an unattached permanent job to which this file is currently attached.
ORI.DATE	Origin date (the date the file was created).
OWNER	For mass storage files the user owns: public (*PUBLIC), permanent (*PERM), local (*LOCAL), or pool (poolname). For mass storage files the user does not own: the user number of the file owner.
TYPE	File type: controllee or data [virtual code (VC) or physical data (PD)].
DT	Device type: mass storage (MS), magnetic tape (NT), or interactive terminal (TE).
FC	File category: system-generated drop file (S), batch file (B), user file (U), user-created drop file (D), and not defined (N).
BT	Blocking type: character count (C), internal (I), record count (K), or non-SIL file (blank).
RT	Record type: ANSI fixed length (F), record mark delimited (R), undefined (U), control word delimited (W), system block (B), or lower CYBER control word (L).
FO	File organization: sequential access (S) or direct access (D).
RP	Retention period (the number of days the file is to be retained). This column is not displayed at a terminal.

INTERACTIVE UTILITY EXECUTION

From an interactive terminal, FILES can be called by name alone; the terminal prompts for parameters.

Information is displayed 15 lines at a time. When output exceeds display size, enter CONTINUE to continue the display or END to terminate the display.

Figure 4-21 shows an example of FILES output from a terminal.

GIVE - CHANGE FILE OWNER

The GIVE control statement changes the ownership of a private or pool file. Figure 4-22 shows the GIVE format. Parameters can appear in any order.

GIVE, { lfn-list }, { U=newown }, PRIVILEGED=x, ACCESS=acs. * POOL=poolname }	
lfn-list	List of 1 through 16 file names, separated by commas, of files whose ownership is to change. The list must not include files with the same name as a public file.
*	Indicates that all local files and attached private permanent files belonging to you are to change ownership.
<u>U=newown</u>	User number of the new owner of private files. If newown is 000000 and the user is privileged, the file becomes public.
<u>POOL=poolname</u>	Name of an existing pool to which the files are given.
<u>PRIVILEGED=x</u>	Indicates whether the file can issue privileged system calls when it is executed. The file must be a controllee file given to a pool by a privileged user. YES Privileged system calls are allowed. NO Privileged system calls are not allowed. If the POOL parameter is omitted, the PRIVILEGED parameter is ignored. If the POOL parameter is specified but the PRIVILEGED parameter is omitted, PRIVILEGED=NO is assumed.
<u>ACCESS=acs</u>	New access permission set (any combination of the following letters without separators). R Read permission W Write permission X Execute permission A Append permission M Modify permission The default access permission sets, if ACCESS is omitted, are shown in table 4-3.

Figure 4-22. GIVE Control Statement Format

The current file owner can give one or more files to any of the following:

- Another user number. The file ownership category remains private, but the user number changes. The file becomes an unattached permanent file belonging to the new owner.
- A pool. The file ownership category changes to pool. No user, including the pool boss, can then access the file without attaching the pool with the PATTACH utility.
- The public file list. The file is given to user number 000000. Only a privileged user can give a file to the public file list.

Only the file owner can change the ownership of a private file. Only the pool boss can change the ownership of a pool file.

You must attach the file before giving it. GIVE cannot change file ownership when the file is attached by another user or opened by a privileged user.

GIVE cannot change the ownership of a public file. If the file has the same name as a public file, GIVE can give the file to a pool or to another user but not to the public file list.

After you give a file, you can no longer access it (unless you give it to a pool to which you have access or unless the new owner grants you access).

When giving a file, either specify a new access permission set for the file or use the former file access permission sets. If you specify the ACCESS parameter, its value is used for each access permission set GIVE defines. If you omit the ACCESS parameter, former file access permission sets are used (except for public files). The set used depends on whether the file is a private file or a pool file before it is given. Table 4-3 lists the old and new file ownership categories of the file and the default access permission sets when the ACCESS parameter is omitted.

Table 4-3. GIVE Default Access Permission Sets

Old File Ownership	New File Ownership	Default Access Permission Sets
Private	Private	The old owner's access permission set becomes the new owner's access permission set.
Private	Pool	The old owner's access permission set becomes the general access permission set for all pool members and the pool boss.
Private	Public	The access permission set contains read and execute permissions only.
Pool	Private	The old pool boss access permission set becomes the new owner's access permission set.
Pool	Pool	The old pool boss access permission set becomes the new pool boss access permission set; the old general access permission set becomes the new general access permission set.
Pool	Public	The access permission set contains read and execute permissions only.

When you give a private file to another user, GIVE immediately changes the stored owner of the file. However, the account identifier associated with the file does not change until the file is referenced by its new owner. The system accounting tables then indicate the total time that the original account owned the file.

A SIL error 1709 is returned by GIVE when you attempt to give a file to a user or a pool and the file size exceeds the maximum allowed file size of the target user or pool.

LABEL - LABEL TAPE FILE

The LABEL control statement specifies the contents of the HDR1 label of a tape file. It stores the HDR1 label specifications for the file specified on the statement in the file index entry of the file. The information in the file index entry is referenced when the file is opened.

If the file is opened for write access, the specified values are written in the new HDR1 label. If the file is opened for read access, the specified HDR1 values are compared with the values in the existing HDR1 label.

Figure 4-23 shows the LABEL control statement format.

LABEL, lfn, FID=fid, RP=rp, FA=x, OFA=x, RT=rt, BT=bt, RLMIN=rlmin, RLMAX=rlmax, PC=x, RMD=x, MPRU=mpru, RPB=rpb, CONVERT=cvt, LPROC=lp, ACCESS=acs, MFN=mfn, FSN=fsn.	
<u>lfn</u>	File name by which the tape file is referenced in the job (one to eight characters). This parameter is required.
<u>FID=fid</u>	File identifier (1 to 17 characters). If FID=fid is omitted and the HDR1 label is written, the file identifier field in the HDR1 label is all blanks. If FID=fid is omitted and the HDR1 label is read, the contents of the file identifier field are not checked.
<u>RP=rp</u>	Retention period in days. The retention period is added to the creation date to determine the expiration date for the file. If RP=rp is omitted, the default value is 30 days.
<u>FA=x</u>	File accessibility character. If the HDR1 label is read, the specified character must match the accessibility character in the label. If the HDR1 label is written, the specified character is written in the label. If FA=x is omitted, the default character is determined by an installation parameter (released value, blank).
<u>OFA=x</u>	Original file accessibility character. If the HDR1 label is to be overwritten, the specified character must match the accessibility character in the existing label. If OFA=x is omitted, the default character is determined by an installation parameter (released value, blank).
NOTE	
When you add a new file to a multifile set, the expiration date of the new file cannot be later than the expiration date of the first file in the multifile set.	

Figure 4-23. LABEL Control Statement Format (Sheet 1 of 3)

Parameters Used to Override REQUEST Statement Values

RT=rt Record type.

- B System block
- F ANSI fixed length
- L CYBER Record Manager (CRM) control word
- R Record mark delimited
- U Undefined
- W Control word delimited

If RT=rt is omitted, the record type specified on the REQUEST statement is used (default, R).

BT=bt Blocking type.

- I Internal
- C Character count
- K Record count

If BT=bt is omitted, the blocking type specified on the REQUEST statement is used (default, C).

RLMIN=rlmin Minimum record length in bytes. If RLMIN=rlmin is omitted, the minimum record length specified on the REQUEST statement is used (default, 1).

RLMAX=rlmax Maximum record length in bytes. If RLMAX=rlmax is omitted, the maximum record length specified on the REQUEST statement is used (default, 0).

PC=x Padding character. If PC=x is omitted, the padding character specified on the REQUEST statement is used (default, blank).

RMD=x Character used as the record delimiter for R format records. If RMD=x is omitted, the record mark delimiter specified on the REQUEST statement is used (default, ASCII US [code #1F]).

MPRU=mpru MPRU size in bytes; used only if the file uses the V tape format. If MPRU=mpru is omitted, the MPRU size specified on the REQUEST statement is used (default, 32768).

RPB=rpb Records per block; used only for the K blocking type. If RPB=rpb is omitted, the records per block value specified on the REQUEST statement is used (default, 1).

Figure 4-23. LABEL Control Statement Format (Sheet 2 of 3)

CONVERT=cvt Data conversion option. If CONVERT is omitted, no conversion is performed and the data is read and written as binary data.

The values for cvt are these:

YES Tape data is read and written as character codes, using the character set specified by the CM parameter.

NO No conversion is performed.

If CONVERT= is specified as YES on the REQUEST control statement, setting either YES or NO on the LABEL control statement has no effect and conversion is done. If CONVERT= is not specified on REQUEST, LABEL can set it to either YES or NO with the expected results.

LPROC=lp Label processing option.

R Read existing labels (verify existing HDR1 label).

W Write new labels.

If LPROC=lp is omitted, the label processing value specified on the REQUEST statement is used (default: R if ACCESS=R or RW, W if ACCESS=W).

ACCESS=acs Data access requested.

R Read access.

W Write access.

RW Read and write access.

If ACCESS=acs is omitted, the access specified on the REQUEST statement is used (default is R).

Parameters for Multifile Sets Only

MFN=mfn Multifile set name (one to eight characters). The multifile set name must be specified on a previous REQUEST statement. If MFN=mfn is omitted, the multifile set name is assumed to be the same as the file name.

FSN=fsn File sequence number (0 to 9999). If FSN=9999 is specified, the file is opened with write access. If LPROC=W is specified, the file is appended to the end of the multifile set. If LPROC=R is specified, the system searches for a file with FSN=fsn. If the file is not found, an error is returned.

If FSN=fsn is omitted, the file is identified by its file identifier as specified on the FID parameter. If neither FSN nor FID is specified, file sequence number 0001 is used.

Figure 4-23. LABEL Control Statement Format (Sheet 3 of 3)

Before a LABEL statement is processed, a REQUEST statement must specify the file name. Unless the operator is asked to mount an unlabeled tape, the REQUEST statement also specifies the tape volumes for the file.

Several data format parameters can be specified on either the REQUEST statement or the LABEL statement or both. If the same parameter is specified on the REQUEST and LABEL statements, the value on the LABEL statement overrides the value on the REQUEST statement.

If the HDR1 label of a file is to contain values other than default values, a LABEL statement is required for the file. A LABEL statement is not required for reading or writing an unlabeled file or a labeled file whose HDR1 label contains only default values.

A LABEL statement is required for reading or writing a file that is a member of a multifile set.

MULTIFILE SETS

A multifile set is a set of tape files. The set can reside on one or more tape volumes. Each file begins with an HDR1 label and ends with an EOF1 label.

A REQUEST statement specifies the name of the multifile set and the tape volumes that belong to the set. A LABEL statement must be specified for each file in the set that you intend to read or write in the job. Each LABEL statement specifies the name of a file and the name of the multifile set to which the file belongs. Later statements reference the file by the file name on the LABEL statement.

A file within a multifile set is identified by its file sequence number and its file identifier. The file sequence number and file identifier of a file are written in its HDR1 label when the file is written.

The data format parameter values on the REQUEST statement for the multifile set apply to all files in the set unless the parameter is also specified on the LABEL statement for the file.

Specify the same LABEL statement parameters for a file in a multifile set as you specify for any tape file.

Writing a Multifile Set

To write a multifile set, specify 1 as the file sequence number for the first file of the set and 9999 as the file sequence number for all subsequent files. If the specified file sequence number is 9999, the actual file sequence number written is the next number in sequence for the set. LPROC=W must be specified.

The LABEL statements for the files in a multifile set can be in any order. No error is returned if a file specified on a LABEL statement is not opened.

For example, suppose a job contains the following statements:

```
REQUEST,FILESET,DEV=NT,VSN=(VOL100,VOL101),AC=W.
LABEL,X,MFN=FILESET,FID=FILE1,FSN=1,LPROC=W.
LABEL,Y,MFN=FILESET,FID=FILE2,FSN=9999,LPROC=W.
LABEL,Z,MFN=FILESET,FID=FILE3,FSN=9999,LPROC=W.
COPY,FIRST,X.
COPY,LAST,Z.
```

The REQUEST statement specifies FILESET as the multifile set name for files written on the tape volumes VOL100 and VOL101. The LABEL statements specify HDR1 labels for files in the multifile set.

The first COPY statement writes the HDR1 label specified on the first LABEL statement and copies any data in file FIRST to the tape file. The file sequence number in the label is 1, and the file identifier is FILE1. The second COPY statement writes the HDR1 label specified on the third LABEL statement and copies any data in file LAST to the tape file. The file sequence number in the label is 2 (the next number in sequence for the set), and the file identifier is FILE3. The second LABEL statement is not used, and its specifications are discarded when the job ends.

Reading a Multifile Set

To read a file in a multifile set, specify its file sequence number on its LABEL statement. The file sequence number specifies the HDR1 label read when the file is opened.

If the file sequence number of the file is not known but its file identifier is known, specify the file identifier and omit the file sequence number on the LABEL statement. If a file identifier is specified, the HDR1 labels in the multifile set are searched until the label containing the specified file identifier is found.

If the HDR1 label of the file contains a nonblank accessibility character, specify the accessibility character on the FA parameter.

For example, suppose a job contains the following statements to read the second file in a multifile set named FILESET.

```
REQUEST,FILESET,DEV=NT,VSN=VOL100,VOL101.  
LABEL,X,MFN=FILESET,FID=SECOND,FSN=2.  
COPY,X,XFILE.
```

The REQUEST statement specifies the multifile set name and the tape volumes to be read. The LABEL statement specifies a local file name for the file, the multifile set name, the file identifier, and the file sequence number. The COPY statement opens file X and copies it to file XFILE. To open file X, it reads the second HDR1 label in the multifile set and checks to ensure that its file identifier is SECOND and that its file accessibility character is blank.

Rewriting Files in a Multifile Set

Files in a multifile set can be rewritten. However, the last file written is always the last file in the set, so when a file is rewritten in a multifile set, all subsequent files to be kept in the set must also be rewritten.

The HDR1 label of a rewritten file need not be rewritten. To verify the HDR1 label but not overwrite the label, specify LPROC=R and ACCESS=W on the LABEL statement.

For example, suppose a multifile set named FILESET has three files. The following statements in a job replace the data in the second file.

```
REQUEST,FILESET,DEV=NT,VSN=VOL100,VOL101,AC=RW.  
LABEL,X,MFN=FILESET,FSN=3.  
LABEL,Y,MFN=FILESET,FSN=2,LPROC=R,AC=W.  
LABEL,Z,MFN=FILESET,FSN=9999,LPROC=W.  
COPY,X,TEMP.  
COPY,NEW,Y.  
COPY,TEMP,Z.
```

The REQUEST statement specifies the multifile set name, the tape volume containing the multifile set, and read and write access for the files in the set. The first LABEL statement is used to read the third file in the set. The second LABEL statement is used to rewrite the data in the second file without rewriting the HDR1 label of the file [the label processing option (LPROC) is read, but the data access (AC) is write]. The third LABEL statement is used to rewrite the third file (data and labels). It specifies file sequence number 9999 because the file is appended to the multifile set.

The first COPY statement copies the third file to a temporary file so that its data is not lost. The second COPY statement overwrites the data in the second file with the data on file NEW. The third COPY statement rewrites the data saved in the temporary file.

LIMITS—LIST USER VALIDATIONS

The LIMITS control statement obtains validation controls and limitations for a user number. You can only obtain the information for the user number executing LIMITS.

The format of LIMITS output follows:

CHARACTERISTICS	
USER NUMBER	User number
DEFAULT ACCOUNT NUMBER	Default account number
ACCOUNT NUMBERS	List of accounts
MASTER ACCOUNT NUMBERS	List of master accounts
DEFAULT PROJECT NUMBER	Default project number
JOB CATEGORIES	Job categories for user
CHARGE STATEMENT REQUIRED	YES/NO
INTERACTIVE ACCESS	YES/NO
PRODUCTION USER NUMBER	*YES/NO
MAXIMUM NUMBER OF	
FILE SIZE	Maximum file size
SECURITY LEVEL	Security level
VALID USER PERMISSIONS	
TAPE ACCESS	
PRIORITY SCHEDULING	
PRIVILEGED	
VARIABLE RATE ACCOUNTING	

An asterisk (*) following the user number indicates it is a production user number. If the user does not have any master accounts, MASTER ACCOUNT NUMBERS is not listed.

Only those permissions which the user is validated for are listed under VALID USER PERMISSIONS. If the user does not have any this section is omitted.

Figure 4-23.1 shows the LIMITS control statement format.

```
LIMITS,LIST=outfile.

LIST=outfile      Name of the optional file on which the output is written.

                  If the LIST parameter is omitted, the default file for batch jobs
                  is OUTPUT, and, for interactive jobs, the output is printed at the
                  terminal.
```

Figure 4-23.1. LIMITS Control Statement Format



LISTAC - LIST ACCESS PERMISSION SETS

The LISTAC utility lists access permission sets.

NOTE

The FILES utility must be used to determine the access permission set of a file you do not own.

LISTAC uses the following conditions to determine the access permission sets it lists for a file.

- File ownership category of the file
- User executing the LISTAC utility
- USER parameter specification

The effect of these conditions is shown in table 4-4.

Table 4-4. Access Permission Sets Listed

File Category	LISTAC Requestor	USER Parameter Specification	Access Permission Sets Listed
Private	File owner	USER=user-list	Individual access permission sets
		USER=GENERAL	General access permission set
		USER=*	All access permission sets
		omitted	Owner's access permission set
Pool	Pool boss	USER=GENERAL	General access permission set
		USER=*	All access permission sets
		omitted	Pool boss access permission set
	Pool member	USER=GENERAL	General access permission set
		USER=*	General access permission set
		omitted	General access permission set
Public	Any user	USER=GENERAL	General access permission set
		USER=*	General access permission set
		omitted	General access permission set

The LISTAC control statement format is shown in figure 4-24. All parameters are optional, and all keyword parameters can appear in any order.

If all parameters are omitted, LISTAC lists the owner access permission set of each private file the user owns.

LISTAC, lfn1-list, PRIVATE= { lfn2-list } , PUBLIC= { lfn3-list } , POOL=pool, lfn4-list,	
USER=user, LIST=outlfn.	
lfn1-list	Private files (1 to 255 file names separated by commas).
PRIVATE= { lfn2-list } *	Private files.
	lfn2-list List of 1 to 255 file names separated by commas.
	* All files belonging to the user.
PUBLIC= { lfn3-list } *	Public files.
	lfn3-list List of 1 to 255 file names separated by commas.
	* All public files.
POOL=pool, lfn4-list	Pool and list of files belonging to to the pool (pool name followed by 1 to 255 file names separated by commas). A pool name specified without a file list requests that access permission sets for all files belonging to the pool be listed.
USER=user	Indicates the access permission set listed.
	For private, pool, or public files:
	GENERAL General access permission set only.
	* All access permission sets.
	For private files only (invalid if pool or public files are also specified on the statement):
	user-list List of user numbers whose individual access permission sets are listed (1 to 16 user numbers, separated by commas).
	Table 4-4 shows the access permission sets listed for each valid USER parameter specification.
LIST=outlfn	Optional name of the file on which output is written.
	If LIST=outlfn is omitted, the default output file depends on whether LISTAC is executed within a batch job or as an interactive task. Batch job output is written on file OUTPUT; interactive task output is displayed at the terminal.

Figure 4-24. LISTAC Control Statement Format

LISTAC OUTPUT

LISTAC lists file information under the following column headings:

<u>Heading</u>	<u>Description</u>
NAME	File name. The file name has an asterisk (*) appended to it if the file is a production file.
OWNER	File ownership: owner user number for a private file, pool name for a pool file, and PUBLIC for a public file.
USER	GENERAL or user number. GENERAL indicates the general access permission set for the file; a user number indicates the user to whom the access permission set applies.
ACCESS	Access permission set: read (R), write (W), execute (X), append (A) and/or modify (M) access permissions, or no access permissions (NONE).

Example 1:

The user with user number 010101 owns files A and B. The user enters the following statement to list the owner access permission set of each file.

LISTAC.

The user receives the following output:

<u>NAME</u>	<u>OWNER</u>	<u>USER</u>	<u>ACCESS</u>
A*	010101	010101	RX
B*	010101	010101	RX

Example 2:

A user with access to pool APOOL enters the following statement to list the general access permission set of public file D and of each file belonging to pool APOOL. The user is not the pool boss for APOOL.

LISTAC,PUB=D,PO=APOOL.

The user receives the following output:

<u>NAME</u>	<u>OWNER</u>	<u>USER</u>	<u>ACCESS</u>
D	PUBLIC	GENERAL	R
PFILE1	APOOL	GENERAL	RW
PFILE2	APOOL	GENERAL	RW

Example 3:

The file owner requests a listing of all access permission sets for private files A and B with the following statement:

LISTAC,A,B,USER=*

The file owner receives the following output:

<u>NAME</u>	<u>OWNER</u>	<u>USER</u>	<u>ACCESS</u>
A	010101	010101	RWX
		GENERAL	RX
B	010101	010101	RWX
		GENERAL	RX
		020202	RW

Example 4:

The file owner requests a listing of individual access permission sets for user numbers 012345 and 012678 for files C and E with the following statement:

```
LISTAC,C,E,U=012345,012678.
```

The file owner receives the following output:

<u>NAME</u>	<u>OWNER</u>	<u>USER</u>	<u>ACCESS</u>
C	010101	012345	RX
		012678	R
E	010101	012345	R

LOAD - GENERATE CONTROLLEE FILE

The LOAD utility generates a controllee file.

A controllee file (also called a virtual code file) is an executable file.

FILES USED TO GENERATE A CONTROLLEE

As shown in figure 4-25, the LOAD utility reads input from object code files and libraries and writes its output on the controllee file and the listing file. The figure also names the default files LOAD uses.

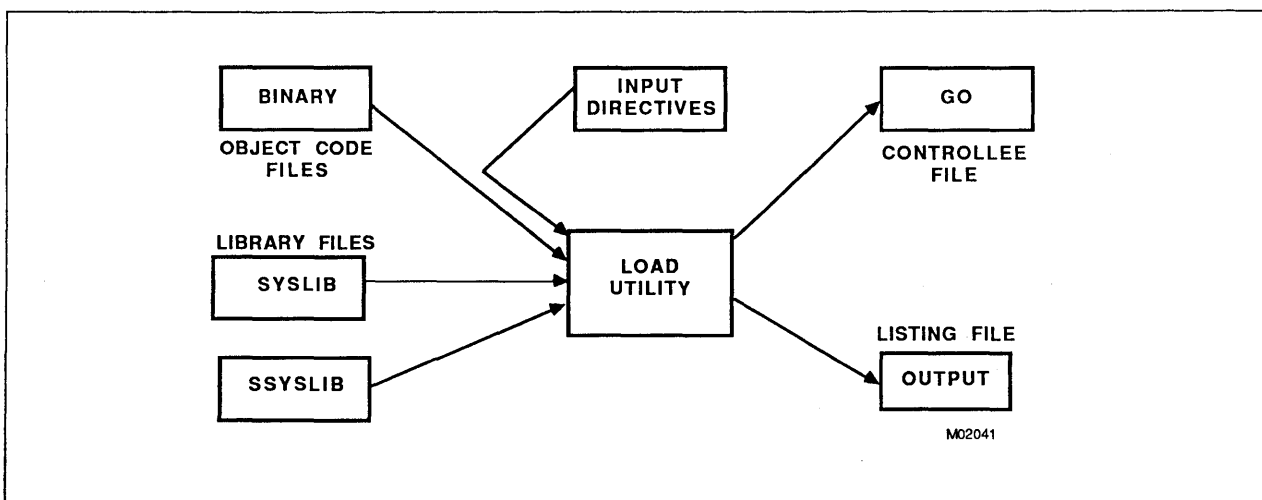


Figure 4-25. Files Used by the LOAD Utility

The number of files used by the LOAD utility cannot exceed 50. These files include the SYSLIB or SSYSLIB file, library files, and object code files.

Library files are generated by the OLE utility described under OLE - Object Library Editor in this chapter. SYSLIB contains all SIL routines and the runtime routines needed to run FTN200 FORTRAN programs. SSYSLIB contains all SIL routines that must always be loaded statically. Whenever the loader builds a dynamic controllee, it always searches SSYSLIB for externals. Routines on SSYSLIB must also reside in SYSLIB.

Object Code Files

An object code file contains object modules generated by a CYBER 200 assembler or compiler. The object code file can be an object file generated by the assembler or compiler or a modmerge file generated by the OLE utility. A modmerge file may contain more than one object code module, but it does not have a directory.

When a LOAD statement references a modmerge file, LOAD loads all modules in the file. Because a LOAD statement can specify only ten object code files, specifying modmerge files enables specification of more code modules for a controllee.

An object code file can be either a local file or an attached private or pool file.

Listing File

The LIST parameter on the LOAD control statement can specify the name of the listing file. If the LIST parameter is omitted, LOAD uses the name OUTPUT. If LIST=0 is specified, no listing is generated.

Having determined the listing file name, LOAD searches for the file. Table 4-5 lists the LOAD processing that results from the file search.

Table 4-5. Results of Listing and Controllee File Searches

If the file	LOAD
Does not exist or is unattached,	requests a new local file.
Exists as a local file,	returns the existing file and requests a new local file.
Exists as an attached private permanent file,	uses the existing private permanent file.

LOAD writes a load map on the listing file, showing the locations of all object modules, databases, and common blocks in the controllee file. If LO=X is specified on the LOAD control statement, the map also includes a cross-reference list of all common blocks and entry points.

LOAD writes the load map with ASCII carriage control characters suitable for printing.

Controllee File

The CNTROLEE parameter on the LOAD control statement can specify the name of the controllee file. If the CNTROLEE parameter is omitted, LOAD uses the name GO.

Having determined the controllee file name, LOAD searches for the file. Table 4-5 lists the LOAD processing that results from the file search. If the file does not exist, LOAD creates the file.

Assuming that the maximum small page size (as set by an installation parameter) is 16 blocks, the default controllee file length is #102 blocks. The controllee file length may be extended to a maximum of #2000 blocks. If a file length of less than #2000 blocks is specified on the CNTROLEE parameter, the file may be extended to #2000 blocks. Files with a length greater than #2000 blocks are never extended by the loader.

When the controllee file is created, its file type must be specified as virtual code.

Read and write access permissions to the controllee file are required. If LOAD creates the controllee file, read, write, execute, append, and modify access permissions to the file are granted.

Controllee File Formats

Figures 4-26 and 4-27 show controllee file formats. The figures show both the virtual bit addresses used when the controllee is mapped into memory and the mass storage word addresses used when it is stored on disk.

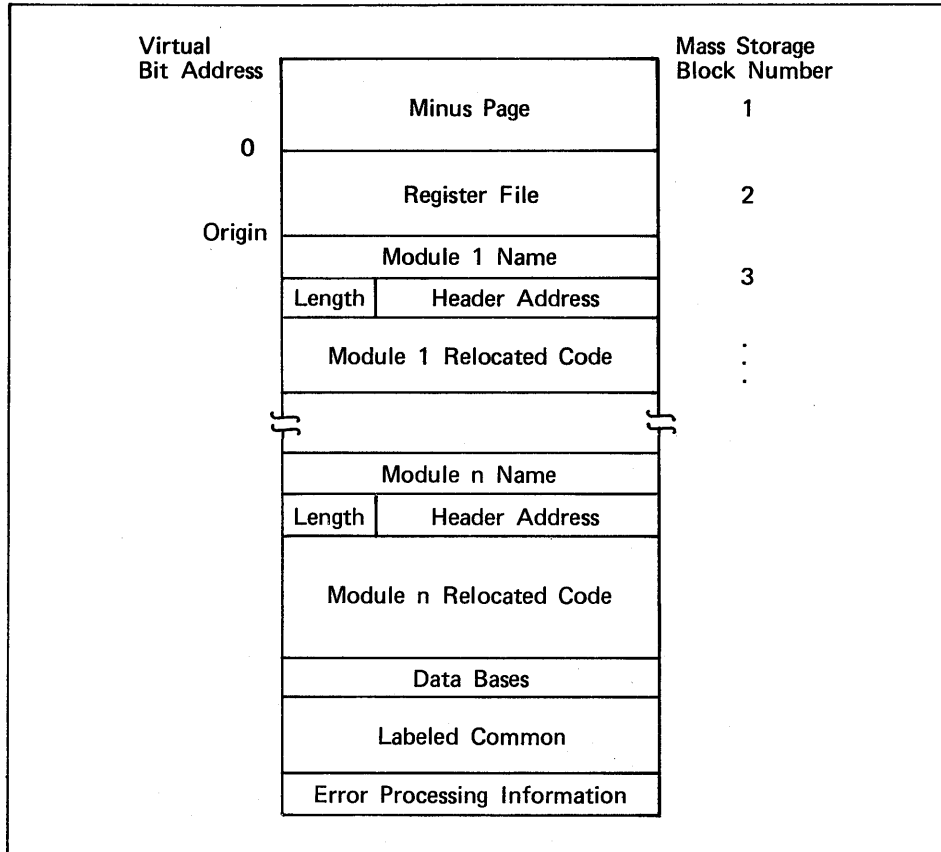


Figure 4-26. Controllee File Format
(Code and Data Bases Separate)

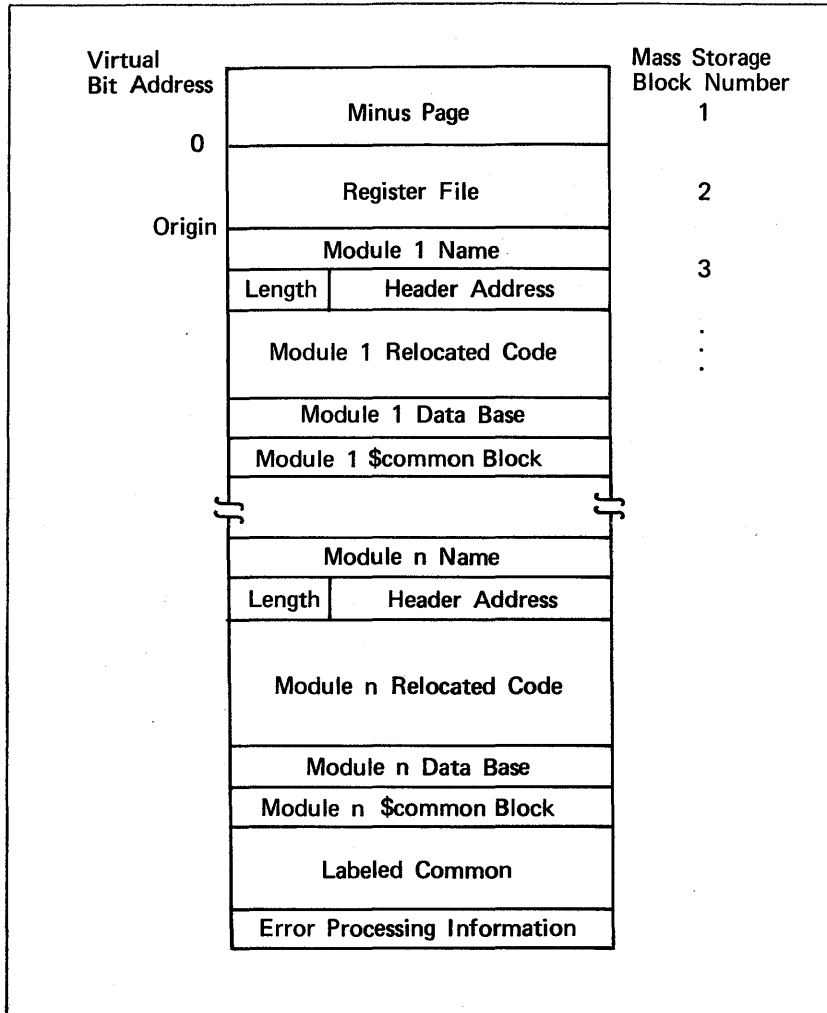


Figure 4-27. Controllee File Format (Data Grouped with Code)

The first one or two blocks of the controllee file are always its minus page. The next block contains its register file. LOAD initializes the system information stored in the minus page and the register file.

LOAD copies object module code to the controllee file, beginning at the default origin address unless the ORIGIN parameter specifies an origin address. If the ORIGIN parameter is omitted but the GRLPALL parameter is specified, the default origin address is #400000 [the first large page (128-block) boundary]. If the ORIGIN and GRLPALL parameters are both omitted, the default origin address is #80000 (the first 16-block boundary).

LOAD copies object modules to the controllee file in the order in which their files are listed on the LOAD control statement. Modules within modmerge files are copied in the order in which they exist on the modmerge file.

Grouping Data With Code

Figures 4-26 and 4-27 show the two possible controllee file formats. The format used is specified by the GDWC parameter on the LOAD statement.

The format shown in figure 4-26 (specified by GDWC=NO) groups all code modules together followed by all data bases. The format shown in figure 4-27 (specified by GDWC=YES) groups each code module with its data base and \$common block, if any. (\$common blocks are used only by the system implementation language, IMPL).

Grouping each data base with its code module can eliminate page faults during controllee execution. This is especially true if the system is using a 4-block or 16-block small page size.

Register File

The register file always occupies bit addresses 0 through #3FFF of the controllee file. When mapped into virtual space, the register file addresses cannot be directly referenced by the controllee.

Specify a character string on the VR parameter to be stored in register #A of the register file. The string can be used as identification in a dump.

LOAD stores the creation date and time for the controllee in registers #B and #C. The date format is eight ASCII characters, mm/dd/yy, representing the month, day, and year. The time format is eight ASCII characters, hh.mm.ss, representing the hour (24-hour clock), minute, and second.

Dynamic Stack

The dynamic stack contains temporary working space used to store the contents of the register file when a subroutine is called. Upon return from the subroutine, the contents of the register file are restored, using the values stored in the stack.

Unless otherwise specified by the DSA parameter, the dynamic stack is allocated following the last virtual address allocated when the task is executed. The dynamic stack address is included in the load map.

SATISFYING EXTERNAL REFERENCES

When loading with LINK=M, LOAD satisfies the unsatisfied external references in the copied object modules. To do so, it searches the directories of the specified library files for a module or entry point name that matches an unsatisfied external reference. When a match is found, LOAD copies the library module to the controllee file. If no match is found in the directories of the specified library files, LOAD searches the directory of the SYSLIB file.

When loading with LINK=D, LOAD searches the directory of the SSYSLIB file satisfying all externals found there. All unsatisfied externals are assumed to be dynamic. Dynamic externals are satisfied at execution time by the linker.

After a library module is copied to the controllee file, LOAD checks all entry points defined for the module to determine whether they match any currently unsatisfied external references. When a match is found, LOAD uses the entry point to satisfy the external reference. Keep this manner of linking in mind when contemplating use of multiple libraries containing modules that have multiple duplicate entry points.

The EQUATE parameter can change external reference linking. If the EQUATE parameter is specified, LOAD performs the requested substitutions of external reference names. The library modules loaded and the common block names used can be changed.

DYNAMIC LINKING USING THE SYSTEM SHARED LIBRARY

Controllees may be built to require or not require the system shared library, SHRLIB, by using the LOAD link option. Controllees built by the loader with LINK=D are completely linked except for the shared SYSLIB module references. During execution of the controllee the shared SYSLIB modules are used for dynamically satisfying SIL routines. If a controllee is partially statically loaded with the shared library active, and then run on a system without the shared library active, the controllee aborts. When the shared library is active and a controllee references a page in the library region that is not locked or mapped, the controllee aborts.

During execution of a dynamically linked controllee, the linker (residing in SHRLIB) performs the functions of LOAD such as data base and common block initialization, relocation, and satisfaction of externals on a dynamic basis when a call to an object module is made.

For loading a controllee that requires SHRLIB, there are various options. All SYSLIB programs can be loaded with the controllee or can be dynamic and linked during execution. Dynamic SYSLIB programs may or may not call entries or reference common blocks in the controllee. The linker may or may not backpatch META programs in the controllee. Backpatching is a modification the linker makes to SYSLIB calls in controllee programs so that only the first call to the SYSLIB routine is dynamic.

Dynamic Linker

The linker does the loading of dynamic modules and gives control to the called module.

The order for satisfying the called module is as follows:

1. If LINK=D was specified, the linker searches only the shared SYSLIB that resides in the shared system library.
2. If LINK=C was specified, the linker first searches the controllee for the entry and then searches the shared SYSLIB.

Loading rules are as follows:

- Dynamically called modules that reference external data cause the module that contains the external data to be loaded and linked by the linker.
- Dynamically called modules that define or try to reduce previously loaded common blocks always use the previous definition.
- The linker ignores the presetting of common blocks by dynamically called modules that preset previously loaded common areas.

Dynamic Execution

Any module that resides in a dynamic library must adhere to the following conventions:

- The module cannot have any external references to code memory sections, data bases, or common within a code msec (for more information on msec, refer to the CYBER 200 Assembler Reference Manual). All external references must be from either the data base or a labeled common block.
- The module must follow the register convention defined in appendix D of volume 2 of this manual.
- The module must not depend on any common to be initialized to zero. Only areas preset by data statements are initialized.
- The length of blank, labeled, and numbered common is established by the first dynamically executed module that defines it.
- If more than one module presets the same common block, presetting is done only by the first module executed. Subsequent presetting of the common block is ignored.
- During dynamic execution, if a module expects a second module to preset a common, the second module must be called before the first module uses that common.

Dynamically Linked Controllees

Controllees that are not completely statically linked should be aware of the following facts relating to dynamic execution.

- Shared SYSLIB modules may vary from one execution to the next.
- Controllees are cautioned to follow calling conventions and be aware of the coding conventions for dynamic modules.
- Controllee files may be smaller, and drop files may be larger.
- Calls from META modules within a controllee file are never backpatched. This means that dynamic calls from META modules go through the linker each time. The LINK=B option can be used to override this restriction.
- IMPL programs should be compiled with OPT=V if possible. This increases dynamic call execution.

PAGE GROUPING

Each grouping parameter (GRLP, GRSP, GRLPALL, GROS, and GROL) specifies a set of object modules or common blocks that LOAD is to process as a group. More than one instance of any grouping parameter can be specified.

Each group of modules or blocks is mapped to a virtual address range. A virtual address range can include more than one page. It begins at the address specified on the grouping parameter.

Grouping parameters do not specify the order of the blocks or modules in a group. The order is determined by the order in which the modules are copied to the controllee file.

All modules or blocks in a group must be of the same type. The three types are code modules, labeled common blocks, and blank or numbered common.

A code module is specified by name, a labeled common block is specified by a name with an * prefix and blank common is specified by an * without a name.

If a virtual bit address on a grouping parameter is specified, ensure that the address has not been allocated. The address must be prefixed by a # character.

Grouping Controllee File Blocks

The GRSP and GRLP parameters group blocks or modules that are stored in the mass storage controllee file.

A group specified on a GRSP parameter is mapped to small pages. A group specified on a GRLP parameter is mapped to large pages.

If the GRLPALL parameter is specified, all modules and blocks are mapped to large pages.

Code modules and labeled common blocks specified on GRSP and GRLP parameters are mapped to the controllee file. Blank common or numbered common blocks are mapped to the drop file.

If no grouping parameters are specified, all object modules, data bases, and common blocks are mapped to small pages.

For example, consider the following grouping parameters on a LOAD control statement.

```
LOAD,GRLP=*BLK1,GRLP=*BLK2,*BLK3,*BLK4,GRLP=*
```

LOAD maps labeled common block BLK1 to a large page and labeled common blocks BLK2, BLK3, and BLK4 to another large page within the controllee file. It maps blank common to a large page within the drop file. (Blank common is always mapped to the drop file.) LOAD maps all other common blocks and modules to small pages.

Grouping Unmapped Blocks

The GROS and GROL parameters group blocks that are not stored in the mass storage controllee file.

The GROS parameter groups blocks to be mapped to small pages; the GROL parameter groups blocks to be mapped to large pages.

Specifying labeled common block groups on the GROS and GROL parameters reduces the size of the stored controllee file because the specified common blocks will be mapped to another file, which will be either the drop file or other mass storage file.

Use of the GROS and GROL parameters could increase the size of the drop file because the grouped labeled common blocks will be mapped by default to the drop file instead of to the controllee file. You can increase the allocated drop file size with the DFL parameter.

Object modules and labeled common initialized with data cannot be grouped with GROS or GROL parameters. Labeled common blocks specified on a GROS or GROL parameter are not preset.

Grouping Parameter Mapping

LOAD does not perform mapping to a file with GROS and GROL as it does with GRSP and GRLP. This mapping is left up to the execution of the controllee or the system at run time.

GROS, GROL, GRSP, and GRLP all align the specified groups or modules on their respective page size boundaries. Specifying GROS or GROL, however, does not determine the unit size of the mapping to a noncontrollee file that includes the drop file. The unit size is either a small or large page.

To create a specific mapping to a noncontrollee file when GROS or GROL was specified for the memory area to map to, the user program calls Q5MAPIN specifying the desired file to map to and the desired unit size of the mapping. Q5MAPIN associates a virtual address range with a mass storage file. If Q5MAPIN is not called before a group or module is accessed, the system creates a small page unit size drop file mapping for the page accessed. Once the system has created a mapping, Q5MAPIN returns an error if it is called with the same virtual range. The user program must call Q5MAPIN for a large page unit size mapping.

When GRSP or GRLP are specified, the LOADER creates a unit size mapping the same size as the type of page boundary to which a common block is aligned.

SPACE INITIALIZATION

LOAD presets labeled common blocks and data base in the controllee file to zero by default or to the value specified by the keyword INITCOM and then initializes data in the blocks as requested by the program. If you specify INITCOM, you may use one of four predefined values to preset common, or use a fifth option that allows you to specify your own pattern. The predefined values are integer zero (all zero bits), floating-point zero, indefinite, and half-word floating-point zero. The predefined constants are represented by the keywords ZERO, INDEF, FPZERO, and HFPZERO. Labeled common blocks specified on GROS or GROL parameters are not preset to zero.

When the program first references blank or numbered common, VSOS provides the program with a page initialized to a memory pattern.

TARGET PAGE SIZE

Specify a target page size for the controllee file with the TSP parameter on the LOAD control statement.

The target page size is the small page size for which LOAD optimizes the controllee structure. If the TSP parameter is omitted, the target page size is the small page size VSOS is currently using.

The small page size VSOS uses is selected during VSOS autoloading. The possible sizes are one, four, and sixteen 512-word blocks.

A controllee optimized for a small page size can be executed when VSOS is executing with that small page size or a smaller small page size. For example, a controllee loaded for the 4-block small page size can execute with a 4-block or 1-block small page size, but it cannot execute with a 16-block small page size.

An attempt to execute a controllee when VSOS is using a small page size larger than the target small page size of the controllee results in the aborting of the job.

The boundary on which LOAD maps a group depends on its type.

LOAD maps each controllee group except blank or numbered common groups to the next available address on a 512-word block boundary. It maps a blank or numbered common group to the next available address on a target page boundary. It maps a group specified by a GROS parameter to the next available address on a selected page size boundary.

If the TSP parameter is not specified, LOAD assumes the controllee will only be run on a system with a small page size equal to the one being used by VSOS at the time the controllee was built. This allows LOAD to make better use of drop file map entries that are generated by LOAD. LOAD allows a drop file map entry to represent up to 32 times the running small page size. If the TSP parameter is specified, a drop file map entry generated by LOAD will never represent more than 32 blocks.

CONTROL STATEMENT FORMAT

Figure 4-28 shows the LOAD control statement format. All parameters are optional. The list of binary object code files must appear before any other parameters. All other parameters can appear in any order.

<pre>LOAD, lfn₁, ..., lfn_n, BINARY=lfn₁, ..., lfn_n, CNTROLEE=lfn/len, DFL=dlen, INPUT=lfn, LIST=lfn/len, LIBRARY=lib₁, ..., lib_n, EQUATE=sub₁, name₁, ..., sub_n, name_n, ENTRY=ept, INITCOM=opt, INITFS=opt, VR=string, ORIGIN=bitadr, TSP=n, GDWC=opt, GRSP=mod₁, ..., mod_n, #bitaddr, GRLP=mod₁, ..., mod_n, #bitaddr, GROS=blk₁, ..., blk_n, #bitaddr, GROL=blk₁, ..., blk_n, #bitaddr, GRLPAL=, LINK=link, DSA=bitadr, LO=X, VALIDATE=Y/N, ULIB=filename, SLIB=filename.</pre>	
<u>lfn_i</u>	List of object code files (1 through 10 file names, separated by commas). If no object code files are listed, LOAD uses file BINARY.
<u>BINARY=lfn_i</u>	List of object code files. This option is the same as the object code list at the beginning of the control statement, except the B= option may appear anywhere within the control statement. If BINARY=lfn _i is omitted and there is no lfn _i list at the beginning of the control statement, LOAD uses the file BINARY.
<u>CNTROLEE=lfn/len</u>	Controllee file. lfn is the name of the file. If lfn is omitted, LOAD writes the controllee on file GO. len is the file length in 512-word blocks. If len is omitted, the default file length is #102 blocks.
<u>DFL=dlen</u>	Number of 512-word blocks in the drop file created when this controllee file is executed. (This value is stored in word #99 of the minus page.) If DFL=dlen is omitted, word #99 of the minus page is zero and the system determines the drop file size. If LINK=D is specified or defaulted, dlen defaults to the length of the controllee plus the length of blank common plus 128 blocks.
<u>INITCOM=opt</u>	If opt is ZERO, labeled common and data base are preset to integer zero; that is, all bits are set to zero. If opt is FPZERO, labeled common and data base are preset to floating-point zero; that is, #8E00000000000000. This pattern also doubles as an illegal instruction. If opt is HFPZERO, labeled common and data base are preset to half-word floating-point zero; that is, #8E0000008E000000. This pattern also doubles as an illegal instruction.

Figure 4-28. LOAD Control Statement Format (Sheet 1 of 6)

If opt is INDEF, labeled common and data base are preset to an indefinite value; that is, #7D1E161C701F1D00. This pattern is:

- An illegal instruction if register 16 contains a value of integer 1
- A 64-bit and 32-bit indefinite floating-point number
- R-type file delimiters of records, groups, and files
- A UNIX string delimiter
- Even parity to cause control word parity errors on V-type files

If opt is #hhhhhhhhhhhhhhhh, labeled common and data base are preset to the user-specified value. The value may consist of 1 to 16 hexadecimal digits.

If INITCOM=opt is omitted, the default value is equivalent to INITCOM=ZERO.

INITFS=opt

If opt is ZERO, free space is preset to integer zero: that is, all the bits are set to zero.

If opt is FPZERO, free space is preset to floating-point zero: that is, #8E00000000000000.

If opt is HFPZERO, free space is preset to halfword floating-point zero: that is, #8E0000008E000000. This pattern also doubles as an illegal instruction.

If opt is INDEF, free space is preset to an indefinite value: that is, #7D1E161C701F1D00. This pattern is:

- An illegal instruction if register 16 contains a value of integer 1
- A 64-bit and 32-bit indefinite floating-point number
- R-type file delimiters of records, groups, and files
- A UNIX string delimiter
- Even parity to cause control word parity errors on V-type files

If opt is #hhhhhhhhhhhhhhhh, free space is preset to the user-specified value. This value may consist of 1 to 16 hexadecimal digits.

If opt is ClF1C, free space is preset to the value #000C1F1C000C1F1C.

If INITFS=opt is omitted, the default value is INITFS=ClF1C.

Figure 4-28. LOAD Control Statement Format (Sheet 2 of 6)

<u>INPUT</u> =lfn	lfn is a local or attached permanent file from which LOAD reads parameters.
	All parameters are allowed on the file except for I= and the object code file list. The I= option may be intermixed with the other options, but it may only be specified once and is processed as if it were specified last.
	The options on the file must be separated by commas, with an optional comma at the beginning of the file. Blanks are allowed after any delimiter just as they are in a control card. EOR is ignored, and a period, right parenthesis, EOG, EOF, or EOI terminates the option.
<u>LIST</u> =lfn/len	lfn is the local or attached permanent file to which LOAD writes the load map. If this parameter is omitted, LOAD writes the map on the local file OUTPUT.
	If LIST=0 is specified, no listing is generated, including the cross-reference listing specified by the LO=X parameter.
	len is the number of 512-word blocks allocated for the file. If len is omitted, #25 blocks are allocated. Unused file space is released at the end of map construction.
<u>LIBRARY</u> =lib _i	List of library files from which LOAD satisfies external references. (OLE creates library files.) If LIBRARY=lib _i is omitted, LOAD searches only file SYSLIB.
<u>EQUATE</u> =sub _i ,name _i	List of external reference pairs. During linking, the second name in each pair is replaced by the first name in the pair.
	Each common block name must be preceded by an asterisk. An asterisk alone indicates blank common.
	The EQUATE parameter cannot change module names.
	This parameter cannot be used to change the name of entry points residing in the system shared library that are to be called dynamically.
	To use this parameter on the system shared library, the controllee must be statically loaded.
<u>ENTRY</u> =ept	Name of an entry point in a loaded module at which execution is to begin (the transfer address). If ENTRY=ept is omitted, Q8MAIN is used.
<u>VR</u> =string	String of one through eight ASCII characters to be stored, left justified and blank filled in register #A. The string cannot contain the characters , .) and blank. If no VR= is specified, the date and time in the form MMDDHHMM is placed in the register.

Figure 4-28. LOAD Control Statement Format (Sheet 3 of 6)

GDWC=opt

Indicates the controllee format used. If GDWC=NO is specified, code modules and data bases are grouped separately (figure 4-26 format). If GDWC=YES is specified, each code module is grouped with its data base and \$common block, if any (figure 4-27 format). If this parameter is omitted, GDWC=YES is used.

GRSP=modlist,
#bitaddr

List of modules or common blocks to be grouped, loaded, and mapped in at the specified bit address.

The names listed for this parameter must all be of the same type (code modules, labeled common blocks, or blank or numbered common). Common block names must be prefixed by an *. An * alone identifies blank common.

If a bit address is specified, it must have a # prefix and be a multiple of the target small page size (a multiple of #8000, #20000, or #80000). If the bit address is omitted, the segment is loaded at the next available block boundary.

GRLP=modlist,
#bitaddr

List of modules or common blocks to be grouped in a segment and loaded at the specified large page address.

The names listed for this parameter must all be of the same type (code modules, labeled common blocks, or blank or numbered common). Common block names must be prefixed by an *. An * alone identifies blank common.

If a bit address is specified, it must have a # prefix and be a large page boundary (a multiple of #400000). If the bit address is omitted, the segment is loaded at the next available large page boundary.

GROS=blklist,
#bitaddr

List of common blocks to be grouped in a segment and loaded at a selected page size boundary. LOAD does not reserve space for the common blocks in the controllee file.

The names listed on an instance of this parameter must be either all labeled common blocks or all blank or numbered common. Common block names must be prefixed by an *. An * alone identifies blank common.

If a bit address is specified, it must have a # prefix; if the specified bit address is not a selected page size boundary, LOAD increases it to the next selected page size boundary (a multiple of #80000). If the bit address is omitted, the segment is loaded at the next available selected page size boundary. When GROS is used more than once, LOAD starts subsequent groups on block boundaries.

Figure 4-28. LOAD Control Statement Format (Sheet 4 of 6)

<u>GROL</u> =comlist, #bitaddr	List of common blocks to be grouped together and loaded at a large page boundary.
	LOAD does not reserve space for the common blocks in the controllee file.
	The names listed for this parameter must be either all labeled common blocks or all blank or numbered common. Common block names must be prefixed by an *. An * alone identifies blank common.
	If a bit address is specified, it must have a # prefix and be a large page boundary. If the bit address is omitted, the segment is loaded at the next available large page boundary (a multiple of #400000).
<u>GRLPALL</u> =Δ	All code modules, data bases, and labeled common blocks are grouped and loaded on large page boundaries. A blank must follow the = sign in the parameter.
	When LINK=D is specified, this parameter has no effect on individual modules that are to be dynamically loaded by LINKER. It does cause the linker to get loading space on the drop file in 128-block increments.
<u>LINK</u> =link	Indicates a complete static load or a partial static load, with all unsatisfied externals to be loaded and executed dynamically by using a linker.
	<p>D Causes LOAD to assume that all unsatisfied externals are to be dynamically loaded and executed. If LINK=D is used when the system shared library is either active or not active, LOAD maps the existing file SHRLIB into its working space and uses it for constructing the controllee. When LINK=D is used, LOAD does not use SYSLIB as the default LIB parameter but uses SSYSLIB instead. A controllee built for dynamic loading and execution does not execute when the system shared library is turned off.</p>
	<p>M Causes LOAD to move all code to the controllee file, doing a complete static load and using SYSLIB as the default for the LIB parameter. LINK=M has the same effect whether the system shared library is active or not.</p>
	<p>C Causes the same as LINK=D but also allows dynamic external modules to call controllee file modules and reference controllee file common blocks.</p>

Figure 4-28. LOAD Control Statement Format (Sheet 5 of 6)

dis

B/CB/BC Causes the same as LINK=D/C but also allows META modules to be backpatched by the linker.

S Causes LOAD to do a complete static link of all externals that would have been dynamic. The code for the externals is not moved to the controllee but is left in the dynamic user or shared library.

When the system shared library is active, the LINK default is D; otherwise, the default is M.

If you expect a dynamic library module to call a controllee file module, you must specify LINK=C. If LINK=D is specified instead, the dynamic library module is linked to a library module instead of to the one being supplied in the controllee file.

If LINK=B/CB/BC is specified and the call follows the proper calling sequences, the linker modifies the META program registers and data (for more information, refer to the Dynamic Execution subsection of the LOAD control statement in this chapter).

DSA=bitadr

Virtual address at which the dynamic stack begins. It must be a page boundary. If DSA=bitadr is omitted, the dynamic stack begins at the last virtual address allocated.

LO=X

Indicates that the load map should include a common block and entry point cross-reference list. All common blocks and entry points are listed alphabetically, with the modules that reference them. If LO=X is omitted, the common block and entry point cross-reference is omitted.

VALIDATE=Y/N

Causes LOAD to validate (make sure they actually exist) all dynamic externals. The externals are only validated to one level; externals referenced by the dynamic externals are not checked. VALIDATE=N is the default. Externals that do not validate are to be considered unsatisfied externals.

ULIB=filename

Causes LOAD to use file filename as a user dynamic library. Externals on this library are satisfied after all libraries specified by the LIB= have been tried. The default is no user dynamic library. ULIB=filename implies LINK=D, unless some other LINK= option is specified.

SLIB=filename

Causes LOAD to use file filename as a system shared library. Externals on this library are satisfied after all LIB= and ULIB= libraries have been tried. If SLIB= is not specified and the system shared library is ACTIVE, LOAD defaults to the active shared library. If SLIB= is not specified and the system shared library is not active LINK=B/C/D is specified, LOAD defaults to the system shared library file. SLIB=filename implies LINK=D.

Figure 4-28. LOAD Control Statement Format (Sheet 6 of 6)

9. Enter either one or more LOAD parameters or a space to specify no more parameter input. Each LOAD parameter must be specified with its keyword (for example, L=PRINTMAP). A comma must separate the parameters if more than one is specified on a line.
10. If you enter another LOAD parameter, LOAD sends the following message:

continue

Enter another LOAD parameter. LOAD repeats the CONTINUE message until you enter a space to end parameter input.

When you enter a space, LOAD begins building the controllee file, using the parameter input.

Figure 4-29 shows an example of interactive prompting for parameter input. LOAD responses are shown in lowercase letters; user entries are shown in uppercase letters. /CR/ indicates a carriage return; Δ indicates that a blank or space should appear. In figure 4-29, the user requests that LOAD use object files XA, XB, and XC, controllee file TONY, and listing file PRINTMAP.

input ?	LOAD request.
XA, XB, XC /CR/	Enter names of files containing code modules.
origin ?	LOAD request.
28000 /CR/	Enter bit address where first module is to be loaded.
entry ?	LOAD request.
Δ/CR/	Indicate default entry point.
any other options ?	LOAD request.
CN=TONY /CR/	Indicate controllee file name.
continue	LOAD response.
L=PRINTMAP /CR/	Indicate listing file.
continue	LOAD response.
Δ/CR/	Terminate parameter input.

Figure 4-29. Example of Interactive LOAD Execution

LOADPF - RELOAD FILES

The LOADPF control statement reloads archived permanent files and queue files. The files must have been archived by the DUMPF utility. For information on the archived file format, refer to the DUMPF statement description earlier in this chapter.

LOADPF can execute concurrently with other tasks, including other LOADPF tasks.

The LOADPF control statement format is shown in figure 4-30. All parameters are optional. All except the first can appear in any order. The first parameter, if specified, must be a list of file names.

The first statement format shown in figure 4-30 is used when reloading from a remote system. The second statement format is used when files are reloaded from CYBER 200 mass storage or tapes.

NOTE

A production file will not retain its production status when reloaded by any user other than the site security administrator. A warning message is output for each file as it is reloaded.

Format for Front-End File Loading

```
LOADPF,lfn-list,USER=userno,POOL=plist,DSET=devset,PACK=packlist,ACCOUNT=alist,  
JCAT=jcatlist,ODSET=oldds,LID=lidlist,SELECT=opts,VERIFY=opt,PURGE=popr,DATE=mmddy,  
TIME=hhmm,NOWNER=nowner,LO=x,LIST=lfn/len,ST=stid,SI=setid, {JCS=strings } .  
                        {INPUT=lfn }
```

Format for CYBER 200 File Loading

```
LOADPF,lfn-list,USER=userno,POOL=plist,DSET=devset,PACK=packlist,ACCOUNT=alist,  
JCAT=jcatlist,ODSET=oldds,LID=lidlist,SELECT=opts,PURGE=popt,VERIFY=opt,DATE=mmddy,  
TIME=hhmm,NOWNER=nowner,LO=x,LIST=lfn/len,DEVICE=device,DENSITY=den,TF=tf,VSN=id-list,  
IU=iu,IRC=irc.
```

lfn-list List of 1 through 128 file names separated by commas. The specified files are assumed to belong to any or all user numbers specified by the USER parameter and/or any or all pools specified by the POOL parameter. If omitted, all files belonging to userno and/or plist are loaded. If SEL=0 is specified, lfn-list identifies the last-group-file(s) of the output-file-family(s) to be reloaded. If SEL=0 is specified and lfn-list is omitted, LOADPF reloads all output-file-families.

USER=userno Private file owners.

For a nonprivileged user:

userno User number of the nonprivileged user.

Figure 4-30. LOADPF Control Statement Format (Sheet 1 of 6)

For a privileged user:

`userno` List of 1 through 128 user numbers separated by commas.

* All file owners, private, pool, and public.

For a system user who has specified the SEL=I or SEL=O parameter(s):

`u-list` List of 1 through 128 user number(s) that queued the files (original owner). If omitted, LOADPF reloads the queue files of all user numbers.

If USER=userno is omitted and the POOL parameter is not specified, LOADPF reloads files belonging to the user number under which LOADPF was run.

POOL=plist List of 1 through 128 pool names separated by commas.

DSET=devset Allows files to be reloaded onto a specific device set. devset is a list of 1 through 128 device set names (DVSTnn) separated by commas. If more than one device set is specified, the device sets will be used in the order they appear in the parameter list, with the first being filled before the next one is used.

PACK=packlist Allows files to be reloaded onto a specific pack. packlist is a list of 1 to 128 pack names (PACKnn) separated by commas. If more than one pack name is specified, the packs will be used in the order they appear in the parameter list, with the first being filled before the next is used.

ACCOUNT=alist For a nonprivileged user, alist is a list of one to seven account identifiers separated by commas. You must be validated for all specified account identifiers in order to archive files with these accounts. For a privileged user, alist is a list of 1 through 128 account identifiers separated by commas. Only files with the specified accounts are archived.

JCAT=jcatlist List of 1 through 64 job categories separated by commas. This parameter is allowed only if SEL=I or SEL=O is specified and applies only to the input queue. If omitted, files belonging to all job categories in the input queue are reloaded.

ODSET=oldds Select files to reload that, when dumped, were resident on the specified device set(s). oldds is a list of 1 through 128 device set names (DVSTnn). Device sets specified do not need to be in the configuration of CYBER 200 executing this LOADPF. If omitted, LOADPF selects files to reload without regard as to their old device set residency.

LID=lidlist List of 1 through 128 destination LIDs for input or output files. This parameter is allowed only if SEL=I or SEL=O is specified. If omitted, all queue files are reloaded regardless of their destination LIDs.

SELECT=opts File characteristics of all files loaded. A file must meet all characteristics specified in order to be loaded. opts can be any combination of the following letters without separators.

A Files accessed on or after the date and time specified by the DATE and TIME parameters. An access is defined as an open but not an attach.

Figure 4-30. LOADPF Control Statement Format (Sheet 2 of 6)

- C Files created on or after the date and time specified by the DATE and TIME parameters.
- I Files in the input queue. Only the system user is allowed to select this option. The I option is mutually exclusive with the PO parameter.
- M Files modified on or after the date and time specified by the DATE and TIME parameters.
- N Reverses the meaning of the A, C, and M options. For example, NC specifies files not created since the date and time specified. This option may appear anywhere in the combination of characters, but it always reverses the meaning of all characters specified.
- O Files in the output queue. Only the system user is allowed to select this option. The O option is mutually exclusive with the PO parameter.
- X Files expired. A file expires when more days have passed since its creation date than the number of days in the retention period for the file.
- R If an existing file has the same name as an archived file, replace the existing file with the archived file. If the file does not exist, it is created.

If SELECT=opts is omitted, LOADPF assumes no options.

PURGE=popt

Purge the pseudo files option indicating whether LOADPF should purge the associated pseudo files of the specified device sets if reload is from mass storage.

YES Purge the pseudo files.

NO Do not purge the pseudo files.

If PURGE=YES is specified and if the LOADPF completes without error, all pseudo files on the specified device set(s) are purged. If PURGE=NO is specified and if the LOADPF completes without error, no pseudo files are purged. If PURGE is omitted, no pseudo files are purged. This parameter is ignored if the reload is not from mass storage.

Figure 4-30. LOADPF Control Statement Format (Sheet 3 of 6)

Verification Parameters

VERIFY=opt Verify the integrity of the archival medium. No reloading of files takes place, but a list file is generated. The files that are verified are specified as if they are to be reloaded. Verification errors are noted in the dayfile and verification is halted after a threshold of errors is reached. The extent of the verification is determined by the opt value as follows:

- Q Quick verification. The archival medium is scanned and selected fields are checked for consistency.
- F Full verification. The quick verification is performed plus the data of each file is read, to ensure it is readable. The length of each file read is compared with the length of the file dumped. This option may take 2 - 30 times as long (or longer) as the Q option depending on the size of the files and the archival medium involved.

If this parameter is not specified, no verification is performed.

DATE=mmdyy Date used by the A, C, and M options on the SELECT=opts parameter. The first two digits of the date indicate the month, the next two digits the day of the month, and the last two digits the last two digits of the year.

If DATE=mmdyy is omitted, LOADPF uses the current date.

TIME=hhmm Time used by the A, C, and M options on the SELECT=opts parameter. hh is the hour, based on a 24-hour clock. mm is the minute in the hour.

If TIME=hhmm is omitted, LOADPF uses midnight.

NOWNER=nowner New owner under which all files selected will be loaded. The NOWNER parameter can be either a user number or a pool name. This parameter is only valid for privileged users.

LO=x Audit information required.

- F Full audit.
- P Partial audit.

If LO=x is omitted, LOADPF writes partial audit information.

LIST=lfn/len Listing file specifications:

- lfn File name (one to eight letters or digits, beginning with a letter). If lfn is omitted, LOADPF uses file OUTPUT.
- len File length in 512-word blocks. If len is omitted, the file length is #40 blocks.

Figure 4-30. LOADPF Control Statement Format (Sheet 4 of 6)

For Front-End File Reloading Only

ST=stid RHF logical identifier of the other system (three ASCII characters). This parameter is required.

SI=setid Set identifier of the archive storage on the other system. This parameter is required. It must be a name of one to six letters or digits beginning with a letter.

 On NOS/BE, the SI parameter is the multifile set name and must be the same name specified on the VSN parameter. For IBM remote hosts, the SI parameter is ignored.

JCS=strings List of 1 to 10 text strings sent to the remote system. Each string must be delimited by double quote (") characters. Strings are separated by commas.

 This parameter cannot be used if the I parameter is used. If both JCS and INPUT are omitted, then I=INPUT is assumed.

INPUT=lfn Name of the CYBER 200 file containing the text string to be sent to the other system. Text strings in the file must appear the same as they would if entered on the JCS parameter without the " delimiters.

 This parameter cannot be used if the JCS parameter is used. If both JCS and INPUT are omitted, then I=INPUT is assumed.

For CYBER 200 File Reloading Only

NOTE

The following parameters are ignored if specified with RHF file archiving parameters.

DEVICE=device Device type from which to load archived files:

 MS Mass storage.

 NT Magnetic tape.

 If DEVICE=device is omitted, the default set by an installation parameter is used.

DENSITY=den Recording density (for tapes only):

 PE 1600 cpi

 GE 6250 cpi

 If DENSITY=den is omitted and DEVICE=NT is specified, the installation-defined default density (released value, 6250 cpi) is used. If the density specified or the default density does not match the density on the tape, processing continues with the density specified on the tape.

Figure 4-30. LOADPF Control Statement Format (Sheet 5 of 6)

<u>TF</u> =tf	Tape format (for tapes only):
	<p>V Variable block size format with block size set to 8K.</p> <p>LB Large block size format.</p>
	<p>If TF=tf is omitted, LOADPF uses the format in which the tape was written. The LB format is the format used by DUMPF prior to the 2.3 release. This parameter is mutually exclusive with DEV=MS.</p>
<u>VSN</u> =id-list	Archive storage device identifiers. VSN=id-list must specify all device sets or tape volumes containing files to be loaded.
	<p>If DEVICE=MS, VSN specifies a list of one through six device sets in the form DVSTnn. nn is #01 through #FF. If DEVICE=NT, VSN specifies a list of 1 through 255 tape VSNs and can be 1 to 6 alphanumeric characters.</p>
<u>IU</u> =iu	Inhibit unload option indicating whether the system unloads a tape volume when the utility is complete. This applies to tape files only.
	<p><u>Y</u> Does not unload tape volume.</p> <p><u>N</u> Unloads tape volume.</p>
	<p>If IU=iu is omitted, N is used.</p>
<u>IRC</u> =irc	Inhibit ring check option. This applies to tape files only.
	<p><u>Y</u> If a tape is already mounted with the ring in, LOADPF will accept the tape with the ring in or the ring out; otherwise, if the tape is not mounted, LOADPF will request the tape without the ring.</p> <p><u>N</u> LOADPF will request the tape without the ring.</p>
	<p>If IRC=irc is omitted, N is used.</p>

Figure 4-30. LOADPF Control Statement Format (Sheet 6 of 6)

RHF RELOADING

If the RHF application program is present on the system, it interprets the ST, SI, JCS, and INPUT parameter specifications. LOADPF uses the ST parameter specification to determine the remote system on which the file copies are stored. The SI, JCS, and INPUT parameter specifications determine where the file copies are stored on the remote system.

LOADPF passes a text string to the remote system to request the file copies. The string is specified on the JCS parameter or in the file specified on the INPUT parameter. The required content of the text string depends on the RHF software in the remote system. For more information, refer to the RHF documentation for the remote system.

RELOADING FROM CYBER 200 MASS STORAGE

When loading files from CYBER 200 mass storage, enter a LOADPF statement on the same user number that the DUMPF was performed and that specifies the DEVICE=MS parameter and the device sets on which files were archived on the VSN parameter.

RELOADING FROM CYBER 200 ON-LINE TAPES

When reloading files archived on CYBER 200 on-line tapes, enter a LOADPF statement that specifies the DEVICE=NT parameter and the VSNs of the tape volumes on the VSN parameter. Optionally, you can specify the recording density on the DENSITY parameter. If the density of the tape does not match that of the density parameter, LOADPF uses the density of the tape and issues an appropriate message.

Files archived on CYBER 200 tapes are stored within a multifile set. For each file reloaded, LOADPF generates the file identifier of its archive file copy, using the file name and owner. (The file identifier format is given in the DUMPF statement description earlier in this chapter.) LOADPF then searches the multifile set for an HDR1 label containing the generated file identifier and reloads the file.

NOTE

If the file was archived by a nonprivileged user, its access permissions must be redefined. The access directory is not saved when a nonprivileged user archives a file.

The TF keyword allows you some control over how LOADPF handles the two formats, LB or V. If this keyword is not specified, LOADPF adapts itself to the format in which the tape was written. If a particular tape format is specified with the TF keyword, LOADPF will only be able to read that particular format. If a format different than that specified with the TF keyword is encountered, LOADPF will abort with an appropriate error message.

USER RELOADING CAPABILITIES

Privileged users can reload any file archived using DUMPF. Privileged users can also reload archived files to a new owner by specifying the NOWNER=nowner parameter. Nonprivileged users can reload their own user number or pool for which they are the pool boss.

If LOADPF cannot restore pool ownership of the file because the pool no longer exists or because the user is not the pool boss for that pool, it reloads the file as a private file.

The information in the first block of each archived file includes the contents of selected fields in the permanent file index entry for the file. If a privileged user archives the file, the first block also contains the following information:

- A copy of the unformatted permanent file index (PFI) entry as it existed after DUMPF opened the file
- A copy of the permanent file index extension entry if an access directory existed for the file

The access fields in the permanent file index entry are updated when DUMPF opens the file. Therefore, the access field information differs, depending on whether the user is or is not privileged. This means that if the A option is specified, the last access date and time used differ, depending on whether the user is or is not privileged.

SPECIFICATION OF THE FILES TO BE RELOADED

The set of files that LOADPF reloads can be specified by the names or the attributes of the files. The set of files must have all the attributes specified. The USER and POOL parameters specify file ownership, and the SELECT, DATE, and TIME parameters can specify file usage and age.

If no file names are specified and the USER and POOL parameters are omitted, LOADPF archives all files belonging to the user number under which LOADPF is executed.

Table 4-6 summarizes the interaction of the USER and POOL parameters.

If a file cannot be reloaded, LOADPF returns an appropriate message and continues processing with the next file. LOADPF output includes a list of the files reloaded.

LOADPF Output

The LO parameter on the LOADPF control statement determines whether LOADPF produces a full or a partial output listing. A full listing produces all of the headings described below, while a partial listing contains only the first 13 headings. A full listing does not exceed 132 characters, excluding the carriage return, and a partial listing does not exceed 80 characters, excluding the carriage return. Dates appear as month, day, and year. Time appears in a 24-hour format. All values are decimal unless noted otherwise.

The following are the column headings used in a full LOADPF listing and the information given under each heading.

<u>Heading</u>	<u>Description</u>
VSN	Volume serial number: this field is printed only the first time a file is loaded from a VSN or when the report goes to a new page.
FSN	File sequence number: hexadecimal count of files loaded.
NAME	File name.
OWNER	File owner: individual user number, public user number (0), or pool name. If SEL=I or SEL=O is specified, then the user number of the original file owner is listed.
TYP	File type: virtual code (VC) or physical data (PD).
FC	File category: batch input file (B), input queue file (I), output queue file (O), user file (U), system-generated drop file (S), or not defined (N).
RT	Record type: ANSI fixed length (F), record mark delimited (R), undefined (U), control word (W), system block (B), or lower CYBER (L).
BT	Blocking type: character count (C), internal (I), or record count (K).
ACS	Access permission set: read (R), write (W), execute (X), append (A) and/or modify (M) permissions; no permissions (NONE); or purge-only (PURGE). LOADPF lists the owner's access permission set for private files and the general access permission set for pool and public files.
EXT	File allocation: segmentable (S) and/or extendable (X).
SL	Security level: 1 through 8.
DEVICE	Device name of mass storage file. An asterisk following the device name indicates that a portion of the file resides on another disk.
DSET	Name of device set.
FLEN	Number of 512-word blocks in file.
FACT	Accounting information.

<u>Heading</u>	<u>Description</u>
DORG	Creation date (date of origin).
TORG	Creation time (time of origin).
DOLA	Date of last file access.
TLR	Time of last file access.
DOLM	Date of last file modification.
TOLM	Time of last file modification.
EXP	Expiration date (creation date plus retention period).

If SEL=I or SEL=O is specified, the TYP column is deleted and the following column headings replace ACS and EXT:

<u>Heading</u>	<u>Description</u>
LID	Destination LID for output and/or input queue files.
JCAT	Job category of input queue files. For all other file types, this field is left blank.

Figure 4-31 shows an example of a LOADPF output listing as produced by the following control statement:

LOADPF,U=*,AC=ACCTN01,ACCTN02,ACCTN03,DEV=NT,VSU=CY2091,CY2088,LO=F,SEL=R.

Table 4-6. Interaction of USER and POOL Parameters for LOADPF

Files Processed	Privileged User						Nonprivileged User			
	No USER		USER=list		USER=ALL		No USER		USER=userno†	
	No POOL	POOL=plist	No POOL	POOL=plist	No POOL	POOL=plist	No POOL	POOL=plist	No POOL	POOL=plist
User private files	X						X			
Listed user private files (or public files if user number 0 is specified)			X	X					X	X
Listed pool files		X		X				X		X
All files regardless of owner (including public and pool files)					X	X				

†Nonprivileged users can specify only their user number.

CYBER 200 LOADPF LOD2219 -USER 14000 06/17/86 13.29.21																					
VSN	FSN	NAME	OWNER	TYP	FC	RT	BT	ACS	EXT	SL	DEVICE	DSET	FLEN	FACT	DORG	TORG	DOLA	TLR	DOLM	TOLM	EXP
CY2091	1	TRACE	10955	PD	U	R	C	XMARW	X	1	PACK3B	DVST3B	16	ACCTN03	028685	931	050885	948	020885	932	031085
	2	CF639B	10955	PD	U	R	C	XMARW	X	1	PACK3B	DVST3B	80	ACCTN03	020685	927	020685	931	020685	927	030885
	3	FT70249B	FTNDROP	PD	U	R	C	XMARW	X	1	PACK3B	DVST3B	80	ACCTN03	102284	1357	112184	1321	102284	1357	112184
	4	MAILIST	9151	PD	U	R	C	RW	X	1	PACK3B	DVST3B	16	ACCTN03	032985	927	060785	1014	060785	1007	042885
	5	V3	9151	PD	U	R	C	XMARW	X	1	PACK3B	DVST3B	16	ACCTN03	072484	1315	052985	846	052985	846	082384
	6	RENAME	BOBP00L	VC	U	U	C	XR	X	1	PACK3B	DVST3B	50	ACCTN03	061385	1250	061385	1251	061385	1250	071385
	7	DELSRC	10011	PD	U	R	C	RW	X	1	PACK3B	DVST3B	16	ACCTN03	052485	1451	052985	1046	052885	923	062385
	8	PFDL	BOBP00L	VC	U	U	C	XR	X	1	PACK3B	DVST3B	82	ACCTN03	051585	1503	061685	2356	051585	1503	061485
	9	DIAG22	9151	PD	U	R	C	XMARW	X	1	PACK3B	DVST3B	16	ACCTN03	031185	1031	061785	812	061285	833	041085
CY2088	A	C2700L	14000	PD	U	R	C	XMARW	X	1	PACK3B	DVST3B	16	ACCTN01	053185	618	053185	618	053185	618	063085
	B	C2700TS	14000	VC	U	U	C	XMARW	SX	1	PACK3B	DVST3B	320	ACCTN01	053185	618	053185	618	053185	618	063085
	C	CEBIN	10955	PD	U	R	C	XMARW	X	1	PACK3B	DVST3B	32	ACCTN03	041285	943	050985	1002	041285	943	051285
	D	MINK	POOLVRF	VC	U	U	C	XMARW	X	1	PACK3B	DVST3B	368	ACCTN03	041285	947	041585	933	041285	947	051285
	E	CG520L	14000	PD	U	R	C	XMARW	X	1	PACK3B	DVST3B	16	ACCTN01	053185	624	053185	624	053185	624	063085
	F	CG520TS	14000	VC	U	U	C	XMARW	X	1	PACK3B	DVST3B	320	ACCTN01	053185	624	053185	624	053185	624	063085
	10	CONNECT	BP22	VC	U	U	C	XR	X	1	PACK3B	DVST3B	64	ACCTN03	061085	1809	061085	1810	061085	1809	071085
	11	DISCONT	BP22	VC	U	U	C	XR	X	1	PACK3B	DVST3B	64	ACCTN03	061085	1809	061085	1811	061085	1809	071085
12	DNAD	9151	VC	U	U	C	XR	X	1	PACK3B	DVST3B	128	ACCTN03	061185	1452	061185	1456	061185	1452	071185	

Figure 4-31. LOADPF Output Example

MFLINK - PERMANENT FILE TRANSFER

The MFLINK control statement transfers a permanent file between VSOS and a remote system (refer to figure 4-32). A permanent file request is embedded in the MFLINK statement. PTF sends the request to the remote system specified on the statement. The request references permanent files residing on the remote system.

The RHF software on the remote system determines the validity of the permanent file request (refer to the RHF documentation for the remote system). The request must use the job control language required by the RHF software on the remote system.

Under certain conditions, MFLINK attempts to recover a transfer that fails before the file has been successfully received by its destination. The transfer is retried from the beginning, and the earlier partial transfer is discarded. In every case, recovery is attempted for batch MFLINK only. The EP or RT parameters can inhibit MFLINK from recovering from some types of failures. Recovery error codes are listed in table 4-7. Refer to appendix B of this manual for an interpretation of the recovery error codes.

Table 4-7. Recovery Error Codes

Recovery Error Code	
8020	8230
8023	8231
8028	8232
8033	8233
8062	8234
8063	8235
8102	8236
8103	8305
8105	8306
8108	8307
8154	8308
8155	8309
8176	8310
8178	8311
8192	8312
8202	8313
8215	8314
8216	8315
8221	8317
8227	8318
8228	8322
8229	8323

For example, the following MFLINK statement contains a request for a copy of an IBM data set. CYBER 200 RHF interprets the control statement and, as specified, sends the text string on the JCS parameter to the system having logical identifier IBM. The RHF software executing on the remote system receives the text string and interprets it as a request. As specified in the text string, it sends a copy of the data set to the CYBER 200 system. CYBER 200 RHF writes the data set copy on file A.

```
MFLINK,A,ST=IBM,JCS="GET,DSN=SEQ.DATA.SET".
```

```
MFLINK,lfn,ST=lid,DD=dd,EP,RT, {JCS="string1","string2",...,"stringn" } .
                               {INPUT=filename }
```

lfn Name of the CYBER 200 file to be copied or to receive the file copy (one to eight alphanumeric characters, beginning with a letter).

If the file is to be copied, it may be a local file, an attached permanent file, a pool file, or a public file. If MFLINK is to transfer data into the file, the file may be a local file, an attached permanent file, or it may not exist. If the file is local and MFLINK is to transfer data into the file, MFLINK returns the file and creates a new file with the same name and with a record type that is compatible with the data format declaration (as it does when the file is nonexistent).

This parameter can be omitted if the specified control statement sequence does not copy a file to or from the CYBER 200 system. If specified, it must be the first parameter.

ST=lid Specifies the logical identifier (LID) of the remote host to which MFLINK is to send the directives record. The LID must be a three-alphanumeric uppercase ASCII character string defined by your site. You must specify the ST parameter on the first, and only the first, MFLINK command of a series of MFLINK commands (an MFLINK session) that are for the same remote host. The ST=lid specification remains in effect during the entire session. A successful completion of the initial MFLINK of a session saves any recovery directives (user or accounting directives) sent to the remote host. Then these recovery directives must not be specified for the remainder of the MFLINK session. On subsequent MFLINK commands in the same session, MFLINK sends the saved recovery directives for you; however, any other MFLINK parameters not specified revert to their default values. The occurrence of the ST=lid parameter on a MFLINK command initiates a new MFLINK session with the specified remote host. Once you have entered an MFLINK command with an ST=lid parameter, you can resume that session at any time during your job by entering an MFLINK command without an ST=lid parameter.

DD=dd Data format declaration. If DD=dd is omitted, each host treats the data transferred as being in the character set used for that system.

C8 Character data from a character set with more than 64 character codes. ASCII separator characters (#1F, #1D, #1E, and #1C) define file structure (SIL R format).

C6 Character data from a character set with 64 or fewer character codes.

Figure 4-32. MFLINK Control Statement Format (Sheet 1 of 2)



US	Binary data with file structure indicated by control words (SIL W format). The logical structure of the file is transmitted via RHF protocol. No data conversion is performed.
UU	Binary data without a logical structure. No file structure or data conversion is performed.
<u>EP</u>	Specifies that RHF should not do error recovery processing if network problems cause a loss of the link during a file transfer. Specifying the EP parameter inhibits the retry process. If EP is omitted, MFLINK attempts to establish a new link and retries the transfer request from the beginning of the request. Partial file transfers are discarded. If EP is specified, it must precede the JCS parameter.
<u>RT</u>	Specifies the real-time action RHF is to take in response to error conditions such as lid disabled, system application limit exceeded, or remote host busy, detected prior to establishment of a link with a remote host. If RT is specified, MFLINK terminates with no additional error recovery. If RT is omitted, the task is held in suspension until the resources become available. In certain conditions, such as lid disabled, when the condition may persist for an indefinite period of time, MFLINK requests via the O display that the system operator RETRY, ABORT, or WAIT (for up to 999 minutes) the task. If RT is specified, it must precede the JCS parameter.
<u>JCS=string_i</u>	One or more text strings sent to the other system. Each string must be delimited by double quote (") characters. (JCS and INPUT are mutually exclusive.) If the JCS parameter is specified, it must be the last parameter on the MFLINK execute line. If both the JCS and INPUT parameters are omitted, then I=INPUT is assumed.
<u>INPUT=filename</u>	Name of the CYBER 200 file containing the text string to be sent to the other system. A text string in the file must appear as it would if entered on the JCS parameter without the double quote delimiters. If INPUT=filename is specified, text strings are read from the specified file. (JCS parameter cannot be specified.) If the keyword INPUT is specified and =filename is omitted, text strings are read from file INPUT. (JCS parameter cannot be specified.) If the INPUT and JCS parameters are omitted, then I=INPUT is assumed.

Figure 4-32. MFLINK Control Statement Format (Sheet 2 of 2)

CHARACTER CODE CONVERSION

The DD parameter on the MFLINK statement is used to specify required character code conversion. If the data format declaration is UU or US, no character code conversion is performed. If the data format declaration is C8, C6, or omitted, character code conversion is performed if necessary.

Character code conversion is necessary when the file transfer is between systems using different character code sets. For example, a CYBER 170 system uses both 6-bit and 12-bit character codes, while a CYBER 200 system uses only 8-bit character codes.

LOGICAL STRUCTURE CONVERSION

RHF copies logical file structure (end-of-record, end-of-group, and end-of-file separators) as specified by the DD parameter on the MFLINK statement. Table 4-8 lists the logical structure conversion RHF performs and the SIL logical record format in which VSOS writes a received file.

Table 4-8. Logical Structure Conversion

MFLINK DD=dd Parameter	RHF Conversion	SIL Format
UU	No logical structure conversion. RHF transfers the file as string of bits terminated by an end-of-information protocol parameter.	Undefined (U) format.
US	Logical structure indicated by file structure control words.	Control word (W) format.
C8	Logical structure indicated by ASCII unit separator, group separator, and file separator characters. The file contains character data from a character set with more than 64 character codes.	Record mark (R) format.
C6	The file contains character data from a character set with 64 or fewer character codes. Logical structure indicated by ASCII unit separators, group separators, and file separator characters. C6 and C8 are treated identically by VSOS but may be treated differently by the remote host.	Record mark (R) format.
omitted	The file is treated by both the sending and the receiving host as being in the native character set of that host. Logical structure indicated by ASCII unit separators, group separators, and file separator characters.	Record mark (R) format.

For FORTRAN data conversion information on sending files to a remote host, refer to appendix E in this manual. Refer to chapter 3 for more information on the operation of RHF permanent file transfers. In particular, note the description of direct access file transfers.

MFQUEUE—EXPLICIT FILE ROUTING

The MFQUEUE control statement submits an output file or a job file to the local system or to a remote system from the CYBER 200 (refer to figure 4-33).

<pre>MFQUEUE,lfn,ST=lid,DD=dd,DC=dc,SAVE, { JCS="string1","string2",...,"stringn" } . INPUT=filename</pre>	
<u>lfn</u>	Name of the CYBER 200 file to be copied to the remote system (one to eight alphanumeric characters, beginning with a letter). This parameter is required.
<u>ST=lid</u>	The RHF logical identifier of the local or remote system (three ASCII alphanumeric characters) to which the file named by lfn is sent. This parameter is optional. If the specified lid is the local host, and DC is either IN or IX, the queue file is submitted as a job to the local host, and the job's output is sent to the remote system specified by the default output LID listed in the Q,0 or H,0 display. If the specified lid is the local host, and DC is neither IX or IN, the queue file is sent to the remote system specified by the default output LID. If ST is not specified and if MFQUEUE is executing from within a batch job, the queue file is sent to the system that submitted the original batch job. If ST is not specified and if MFQUEUE is performed interactively, the queue file is sent to the remote system specified by the default output LID.
<u>DD=dd</u>	Data format declaration. RHF performs any conversion necessary to maintain the specified data format and job structure (see Character Code Conversion and Logical Structure Conversion in the description of MFLINK). If DD=dd is omitted, each host treats the data transferred as being in the character set used for that system.
C8	Character data from a character set with more than 64 character codes. ASCII separators (#1F, #1D, #1E, and #1C) define file structure (SIL R format).
C6	The file contains character data from a character set with 64 or fewer character codes. C6 and C8 are treated identically.
US	Binary data with structure indicated by control words (SIL W format). The file structure is transmitted via RHF protocol. No data conversion is performed.
UU	Binary data without a file structure. No file structure or data conversion are performed.

Figure 4-33. MFQUEUE Control Statement Format (Sheet 1 of 3)

DC=dc

Optional disposition code. If DC is not specified, disposition of LP (print) is assumed.

IN Route the file to the input queue of the remote system specified by the ST parameter. The generated output files are returned as follows:

- If MFQUEUE is executing from within a batch job, the output is returned to the system that submitted the original batch job.

If the system that submitted the original batch job was the local system, the output is returned to the remote system specified by the default output LID listed in the Q,0 or H,0 display.

- If MFQUEUE is performed interactively, the output is returned to the remote system specified by the default output LID.

IX Route the file to the input queue of the remote system specified by the ST parameter.

If the LID specified by the ST parameter is a remote system, the generated output is disposed of by the remote system. If the LID specified by the ST parameter is a local system, the generated output is sent to the remote system specified by the default output LID.

LP Route the file to be printed on the remote system.

CP Route the file to be punched in hollerith code to the remote system.

P8 Route the file to be punched in 80 column binary to the remote system.

PB Route the file to be punched in checksummed data form to the remote system.

SP Route the file as special output to the remote system.

If the LID specified by the ST parameter is the local system, the file is sent to the remote system specified by the default output LID.

Figure 4-33. MFQUEUE Control Statement Format (Sheet 2 of 3)

SAVE	Optional standalone parameter to save the lfn. If SAVE is specified, MFQUEUE will copy lfn to a local file and the copied file is then sent to the remote system. Thus, lfn remains local. (This behavior is identical to what happens if lfn is an attached permanent file.) If SAVE is omitted, and if lfn is a local file, lfn is sent to the remote system. If SAVE is specified, it must precede the JCS parameter.
<u>JCS</u> =string ₁	One or more text strings sent to the other system. Each string must be delimited by double quote (") characters.
<u>INPUT</u> =filename	Name of the CYBER 200 file containing the text string to be sent to the other system. The text string must appear as it would if entered on the JCS parameter without the double quote delimiters. If INPUT=filename is specified, text strings are read from the specified file. (The JCS parameter cannot be specified.) If the keyword INPUT is specified and =filename is omitted, text strings are read from file INPUT. (The JCS parameter cannot be specified.) If the INPUT parameter is not specified, text strings from JCS parameter are used. If neither JCS nor INPUT is specified, no explicit text is sent to the other system.

Figure 4-33. MFQUEUE Control Statement Format (Sheet 3 of 3)

To explicitly route an output file by using the MFQUEUE statement, specify the name of the output file, the identifier of the system to which the file is sent, and any job control statements that may be required by the other system.

If a job file is being sent, the MFQUEUE statement specifies the name of the file, the identifier of the system to which the file is sent, and any job control statements that may be required by the other system. To determine the appropriate job control statements, refer to the RHF documentation for the remote system. The file data must have the job structure required by the remote system. Output from job execution is routed to an output device on the remote system; it is not returned to the CYBER 200 system on which the job originated.

Refer to appendix E in this manual for FORTRAN data conversion information on sending files to a remote host.

A successful completion of MFQUEUE results in the following message:

lfn MFQUEUEED - JDN = jdn

lfn Logical file name of the file being MFQUEUEED.

jdn Job descriptor number assigned by VSOS to the MFQUEUEED file while in the output queue.

NORERUN - SET NORERUN STATUS

The NORERUN control statement changes the default status for the job from rerun to norerun, so that when the system is reauto-loaded after a system failure, the batch input file is destroyed.

The NORERUN control statement is valid only within a batch job. It is executed directly by the batch processor.

Figure 4-34 shows the NORERUN format. An example of NORERUN use is shown in figure 4-35.

```
NORERUN.
```

Figure 4-34. NORERUN Control Statement Format

```
RESOURCE,TL=8.  
.  
.  
.  
COMMENT.           Job reruns if the system fails here.  
NORERUN.  
.  
.  
.  
COMMENT.           Job does not rerun if the system fails here.  
RERUN.  
.  
.  
.  
COMMENT.           Job reruns if the system fails here.  
.  
.  
.
```

Figure 4-35. NORERUN/RERUN Example

OLE - OBJECT LIBRARY EDITOR

The OLE control statement generates any of the following file types:

- A library file. A library file contains code modules and a directory of module names and entry points. The LOAD utility uses the library files specified on its LIBRARY parameter to satisfy external references. It copies the library modules needed to satisfy external references and to obtain the entry address specified on its ENTRY parameter.
- A modmerge file. This file is a collection of modules without a directory. When a LOAD statement references a modmerge file, LOAD copies all modules in the file. Because a LOAD statement can specify only ten object code files, specifying modmerge files enables specification of more code modules for a controllee.
- A dynamic library. A dynamic library file contains code modules and a directory of entry points. The file resides in user space below #800000000000. This library may be dynamically or statically linked by LOAD for any controllee loaded with the dynamic library specified. Statically linking to a dynamic library can mean the code is not moved to the controllee but is left in the dynamic library. This means the dynamic library must be mapped in by the system whenever the controllee executes. Dynamic linking to a user dynamic library means that the linker links any modules that are dynamically called during the execution of the controllee.

OLE can also list modules and characteristics of the modules on library or modmerge files without creating a new file.

As many as 50 input files can be specified on the OLE statement. An OLE input file must be a library file, a modmerge file, or an object code file generated by a CYBER 200 compiler or assembler. An input file can be a local file, an attached private or pool file, or a public file. The total number of modules and entry points cannot exceed 3000.

Specify the name of the library or modmerge file on the NEWLIB or MODMERGE parameter and the name of the listing file on the OUTPUT parameter. OLE searches for the specified file. The following is the result of the search.

<u>If the file</u>	<u>OLE</u>
Does not exist or is unattached,	requests a new local file.
Exists as a local file,	returns the existing file and requests a new local file.
Exists as an attached pool file or public file with the same name,	requests a new local file.
Exists as an attached private permanent file,	uses the existing private permanent file.

Figure 4-36 shows the OLE format. Parameters can appear in any order, but subparameters cannot be separated.

OLE, INPUT=lfm-list, { NEWLIB=liblfm MODMERGE=modlfm }, OMIT=sfn, mod-list, SELECT=sfn, mod-list, LO=opt, OUTPUT=lfm/len, DLIB=lfm, ORIGIN=bitaddr.	
<u>INPUT=lfm-list</u>	List of 1 through 50 file names, separated by commas, whose modules are to be written to the file specified by the NEWLIB or MODMERGE parameter. If the INPUT parameter is omitted, only the parameters LO and OUTPUT are valid.
<u>NEWLIB=liblfm</u>	Name of file that is to contain the new library being created. The name must consist of one through eight letters or digits, beginning with a letter, and can duplicate a file name specified with the INPUT parameter. If the NEWLIB, MODMERGE, and DLIB parameters are omitted and INPUT is specified, default is NEWLIB.
<u>MODMERGE=modlfm</u>	Name of file that is to contain the modmerge file. The name must consist of one through eight letters or digits, beginning with a letter. The name can be the same as the name of a file specified with the INPUT parameter. No default name exists.
<u>OMIT=sfn, mod-list</u>	List of modules (mod-list) on file sfn to be omitted from the library or modmerge file. One OMIT parameter can be specified for each input file that does not have a SELECT parameter specified.
<u>SELECT=sfn, mod-list</u>	List of modules, mod-list, on file sfn to be included in the library or modmerge file. The SELECT parameter can be specified for each input file that does not have an OMIT parameter specified.
<u>LO=opt</u>	Listing option: lfm-list List of file names, separated by commas, whose contents are to be listed. The module name, length, creation date, and entry point name for each file are listed. OLE always lists the contents of the library or modmerge file. 0 Suppress all listings. If the LO=opt parameter is omitted, only the library or modmerge file is listed. Output appears on the file specified by the OUTPUT parameter.
<u>OUTPUT=lfm/len</u>	File to which listing is written: lfm Name of file. Must consist of one through eight letters or digits, beginning with a letter. When the OUTPUT parameter is omitted, default is OUTPUT. len Number of 512-word blocks in file. When /len is omitted, default is #10.

Figure 4-36. OLE Control Statement Format (Sheet 1 of 2)

DLIB=lfn

OLE is to construct a user dynamic library named filename. DLIB is mutually exclusive with the NEWLIB and MODMERGE parameters. OLE checks all modules going to the library to make sure they can reside in a dynamic library.

ORIGIN=bitadr

Virtual bit address at which the dynamic library resides when the controllee executes. This address must be at a lower number than the address #7FFF00000000 at which DEBUG resides. ORIGIN may be specified only when DLIB is specified. ORIGIN defaults to #400000000000.

Figure 4-36. OLE Control Statement Format (Sheet 2 of 2)

During OLE execution, modules are copied to the new file in the order in which they are read. Input files are read in the order in which they are listed on the INPUT parameter. The SELECT and OMIT parameters provide for selective inclusion or exclusion of modules from the new file. If duplicate module names exist in the input files, only the first module encountered is copied to the new file.

PACCESS - AUTHORIZE POOL ACCESS

The PACCESS utility establishes the list of users authorized to attach a pool. Only the pool boss can execute PACCESS. The control statement format is shown in figure 4-37.

PACCESS,poolname,USER= $\left\{ \begin{array}{c} \text{ulist} \\ * \end{array} \right\}.$	
poolname	Name of an existing pool for which the user is pool boss
<u>USER</u> = $\left\{ \begin{array}{c} \text{ulist} \\ * \end{array} \right\}$	Users who are granted access to the pool
	ulist List of 1 through 32 user numbers separated by commas
	* All users

Figure 4-37. PACCESS Control Statement Format

PASSWORD - CHANGE USER PASSWORD

The PASSWORD utility changes the password associated with a user number. Subsequent system accesses must specify the new password.

VSOS stores user passwords in coded form so that no one can list them. You must, therefore, keep a record of the password; if you forget it and have kept no record of it, ask the system operator to change the password. The system operator can change passwords but cannot list current passwords.

When executed, the PASSWORD statement is written in the job and system dayfiles, but the passwords specified on the statement are suppressed and are not written in either dayfile.

A password can contain any ASCII character. However, if the password contains any of the following characters or an embedded blank, it must be enclosed in double quote characters ("").

, " . / & # () =

To specify a quote character in a password, specify two quote characters for each quote character in the password. To specify a blank password, enclose one or more blank characters in quotes.†

For example, the following PASSWORD statement changes a blank password to "=":

```
PASSWORD," ","="="="
```

The PASSWORD control statement format is shown in figure 4-38. You must specify the parameters in the order shown, and both parameters are required. Either a comma or a left parenthesis can follow the PASSWORD verb; either a period or right parenthesis can terminate the statement.

PASSWORD,oldpw,newpw.	
oldpw	Old password (one to eight characters)
newpw	New password (one to eight characters)

Figure 4-38. PASSWORD Control Statement Format

†Site personnel determine during system installation whether user password entry is required or optional. If password entry is required, a blank password is invalid. If password entry is optional, a blank password is valid.

PATTACH - ATTACH A POOL

The PATTACH utility attaches a pool to a job. Interactive tasks or batch jobs for a given user number can access the files belonging to the attached pool.

A job can have up to four pools attached at a time. This includes the system pool if one exists. (The system pool is automatically attached to each active job.)

The access allowed to each pool file depends on the ACCESS parameter specified when the file was given to the pool. Refer to output from the AUDIT or FILES utilities to determine the access permissions.

The control statement format is shown in figure 4-39.

```
PATTACH,poolname.
```

```
poolname           Name of an existing pool for which you are granted  
                   access.
```

Figure 4-39. PATTACH Control Statement Format

PCREATE - CREATE A POOL

The PCREATE utility defines a pool entry in the pool list. If you are the user defining the pool, you become the pool boss. Only the pool boss can perform the following functions for the pool.

- Give files belonging to the pool to another owner
- Purge files belonging to the pool
- Grant other users access to the pool
- Remove user authorization to access the pool
- Destroy the pool

The control statement format is shown in figure 4-40.

PCREATE,poolname.	
poolname	Name of the new pool (one to eight letters or digits, beginning with a letter)

Figure 4-40. PCREATE Control Statement Format

PDELETE - REMOVE USER ACCESS TO A POOL

The PDELETE utility removes user numbers from the list of authorized users. Only the pool boss can execute PDELETE for the pool.

If you, as a user, attached the pool prior to the pool boss' executing a PDELETE control statement to remove your access to the attached pool, the removal of your access becomes effective after you detach the pool.

If all users were granted access to the pool (universal access), PDELETE cannot remove the access permission of individual user numbers. The pool boss must first remove access for all users and then grant access to individual user numbers.

The control statement format is shown in figure 4-41.

PDELETE,poolname,USER=	$\left\{ \begin{array}{c} \text{ulist} \\ * \end{array} \right\} .$
poolname	Name of an existing pool for which the user is pool boss.
<u>USER</u> =	$\left\{ \begin{array}{c} \text{ulist} \\ * \end{array} \right\}$
	Users whose access permission to the pool is removed.
	USER=* must be specified if all users were granted access to the pool.
	ulist List of 1 through 16 user numbers separated by commas
*	All users.

Figure 4-41. PDELETE Control Statement Format

PDESTROY - DESTROY A POOL

The PDESTROY utility destroys a pool. Only the pool boss can execute PDESTROY for a pool.

A pool cannot be destroyed while either the pool boss or a pool member has the pool attached. To determine whether any users have the pool attached, the pool boss must enter a PFILES statement to see whether the user count is zero.

A pool cannot be destroyed while the pool owns files. To list the files belonging to a pool, the pool boss uses the FILES utility. To remove files from a pool, he or she either purges the files or gives them to another owner.

The control statement format is shown in figure 4-42.

PDESTROY,poolname.	
poolname	Name of an existing pool for which the user is pool boss

Figure 4-42. PDESTROY Control Statement Format

PDETACH - DETACH AN ATTACHED POOL

The PDETACH utility detaches an attached pool from a job. A pool may not be detached by PDETACH if there are any open files in the pool.

The control statement format is shown in figure 4-43.

PDETACH,poolname.	
poolname	Name of pool from which user is to be detached

Figure 4-43. PDETACH Control Statement Format

PERMIT - CHANGE ACCESS PERMISSION SET

The PERMIT utility can change an existing access permission set.

Only the file owner can change an access permission set of a private file. Only the pool boss can change the access permission sets of a file belonging to a pool. Only a privileged user can change the access permission set of a public file.

Attach a private file to change its access permission sets. It is not necessary to attach the pool to change an access permission set of a file belonging to the pool.

The access permission changes specified on a PERMIT control statement are only effective for subsequent attachments of a private file (or subsequent openings of public or pool files). The changes do not affect current attachments or openings.

Do not give write access permissions (W, A, or M) to a production file (refer to chapter 7 of the Installation Handbook for further details) or a fatal error results and the system displays a fatal error message. Also, do not give write access permissions to any dayfile. Any write permissions you specify will be ignored and a warning message displayed.

The PERMIT control statement format is shown in figure 4-44. Specify the permanent file and the new access permission set. Optionally, you can set the type of access permission.

PERMIT, { lfn
 POOL=poolname,lfn } ,USER=user,ACCESS=acs.
 PUBLIC=lfn

lfn Attached private file name.

POOL=poolname,lfn Pool name and pool file name.

PUBLIC=lfn Public file name.

USER=user Indicates the access permission sets changed:

 user-list The individual access permission sets for the
 specified user numbers (1 to 16 user numbers
 separated by commas). This option is valid only
 for a private permanent file.

 GENERAL Everyone but the owner is given the permission
 set specified by the AC parameter.

 * All access permission sets.

If USER=user is omitted, the access permission set changed
depends on the file category as follows:

 Private Owner's access permission set

 Pool Pool boss's access permission set

 Public General access permission set

ACCESS=acs New access permission set. ACCESS=NONE indicates no access
 permissions. Otherwise, access permissions are indicated by a
 string composed of one or more of the following letters:

 R Read permission

 W Write permission

 X Execute permission

 A Append permission

 M Modify permission

The ACCESS parameter is required.

Figure 4-44. PERMIT Control Statement Format

PFILES - LIST POOL INFORMATION

The PFILES utility produces a list of information about pools in the system. Specific information listed depends on the parameters selected. The format is shown in figure 4-45.

PFILES displays pool information in alphabetical order by name.

If information being reported exceeds the size of the display screen of an interactive terminal, the last line of the display instructs you to enter MORE or YES to continue the display.

```
PFILES, { poolname
          { USER= { u-list
                  *
                }
            }
        } .
```

poolname Name of existing pool for which authorized users are to be listed.

USER= { u-list
 { *
 } Identification of pool bosses:

 u-list List of 1 through 16 pool boss user numbers separated by commas. All pools belonging to each pool boss are listed by name, along with the number of users currently attached to the pool.

 * All pool names are to be listed along with the pool boss and number of users currently attached to each pool.

Figure 4-45. PFILES Control Statement Format

Figure 4-46 shows output examples. The USER COUNT column lists the number of users to which the pool is attached.

```
PFILES(USER=*)
```

POOL NAME	POOL BOSS	USER COUNT
DEMO	300299	0
STESTS	333333	0
TESTPOOL	333322	2

```
PFILES(DEMO)
```

USERS GRANTED ACCESS TO POOL

```
300045 300047 300034 300089
000022
```

Figure 4-46. PFILES Sample Output

PROCEED - SET ABNORMAL TERMINATION PATH

The PROCEED control statement establishes the point at which the batch processor continues job processing after a task returns an abnormal termination code and works the same as the EXIT statement.

If the PROCEED statement is encountered during normal job advancement to the next control statement, job processing processes the PROCEED statement as if it were a comment. An EXIT statement terminates processing in this case.

The PROCEED control statement is valid only in a batch job. It is executed directly by the batch processor.

The PROCEED control statement format is shown in figure 4-46.1. More than one PROCEED control statement can appear in a job.

```
PROCEED.
```

Figure 4-46.1. PROCEED Control Statement Format

PURGE - DESTROY PERMANENT OR POOL FILES

The PURGE control statement destroys one or more permanent mass storage files. It releases the file space for reassignment.

A PURGE statement can destroy any of the following files:

- Up to 16 private permanent files owned by the caller
- Up to 16 pool files belonging to a pool to which the caller is attached

PURGE cannot destroy a file that is currently attached or opened by a privileged user.

To purge a pool file, the following conditions must be met.

- You must be the pool boss of the pool to which the file belongs.
- You must attach the pool owning the file to be purged before any other pool that owns a file with the same name is attached.

To destroy a permanent file accessible through RHF, execute an MFLINK statement as described in this chapter.

The PURGE control statement format is shown in figure 4-47.

<code>PURGE,lfn-list,POOL=poolname.</code>	
<code>lfn-list</code>	A list of 1 through 16 CYBER 200 private permanent files separated by commas or a list of 1 through 16 CYBER 200 pool files to be purged. Private and pool files cannot both be specified in a single list.
<code>POOL=poolname</code>	The pool to which the files named in lfn-list belong. The user must be the pool boss and must be attached to the pool. If POOL=poolname is omitted, the files named in lfn-list must be private files.

Figure 4-47. PURGE Control Statement Format

Q - LIST JOB STATUS

The Q control statement can list status information for any or all of the following file or task categories:

- Batch input files in the input queue
- Executing tasks
- Output files

The Q utility executes in either privileged mode or nonprivileged mode. The site determines the mode used. If Q is operating in nonprivileged mode, a nonprivileged user receives status information only for the batch job or files belonging to him or her.

	<u>Privileged Mode</u>	<u>Nonprivileged Mode</u>
Privileged users	Q lists tasks or files regardless of owner.	Q lists tasks or files regardless of owner.
Nonprivileged users	Q lists tasks or files regardless of owner.	Q lists only the tasks or files owned by the user.

If the Q statement specifies a file name, the Q utility searches the specified queues for the file and displays the file entry if it is found. If Q does not find the file entry, it displays the following message and terminates.

```
lfn NOT FOUND
```

If no file name is specified on the Q statement, the Q utility lists all entries in the specified queues.

If the specified file name is too long, Q displays the following message and then terminates.

```
INVALID JOBNAME
```

If executed within a batch job, the Q utility sends status information to the job dayfile.

At an interactive terminal, Q displays 15 lines of information at a time. If more information is available, Q displays the following message:

```
ENTER "C" TO CONTINUE "END" TO TERMINATE
```

To display additional information, enter CONTINUE; any other entry terminates the utility.

Figure 4-48 shows the Q control statement format.

Q, $\left\{ \begin{array}{l} \text{INPUT} \\ \text{EXECUTE} \\ \text{OUTPUT} \\ \text{jdn} \\ * \end{array} \right\}$, jobname.

I NPUT Lists status information for batch input files.

E XECUTE Lists status information for executing tasks.

O UTPUT Lists status information for output files.

jdn Job descriptor number. When this parameter is specified, the Q statement lists only the status information for those batch input files, executing tasks, and output files that have the specified jdn. This parameter is mutually exclusive with jobname.

* Lists status information for batch input files, executing tasks, and output files.

jobname Job name of the queue file. When this parameter is specified, the Q statement lists only the status information for those queue file(s) that have the specified jobname. This parameter is mutually exclusive with jdn.

Figure 4-48. Q Control Statement Format

INPUT QUEUE STATUS

Q returns input queue information as a table with the following column headings:

<u>Heading</u>	<u>Description</u>
JDN	Job descriptor number associated with the job.
JOBNAME	The eight-character batch input file name derived from the job name supplied on the job card.
USER	User number to which the file belongs (six decimal digits).
LID	Logical mainframe identifier of the remote system to which the job will return after processing (three characters).
PRI	Priority (two decimal digits) and subpriority (three decimal digits).
STATUS	Job status indicating why the input queue manager has not selected the job for execution. The possible job status identifiers are listed in table 4-9.
TL	Time limit for the job in system seconds (decimal integer).
WS	Working set in 512-word blocks (decimal integer).
LP	Large page limit (decimal integer).
NT	Maximum number of tape units that could be assigned to the job at one time, as specified on the job RESOURCE statement (decimal integer). If there are none, the field is blank.
JCAT	Job category identifier (one to eight characters).

If input queue status is requested but no input queue files exist to be listed for the user, Q sends the following message to the job dayfile or interactive terminal.

NO JOBS IN INPUT QUEUE

Table 4-9. Input Queue Status Identifiers

Status	Meaning
HELD	The operator is holding the job in the input queue.
JMAX	The number of executing jobs for the user or the job category is at maximum.
JOFF	The job category is disabled.
MXLP	The job's large page limit exceeds the maximum large page limit for the job category. The operator has lowered the limit.
MXMO	Scheduling the job would exceed the maximum memory overcommitment.
MXNT	Scheduling the job would exceed the maximum tape drives commitment.
MXRR	Scheduling the job would exceed the maximum rerun time.
MXTL	The job time limit exceeds the maximum time limit for the job category. The operator has lowered the limit.
MXWS	The job working set exceeds the maximum working set size limit for the job category. The operator has lowered the limit.
NEW	The input queue manager has not yet processed the job.
SPRZ	The job subpriority is zero because more jobs than the maximum allowed have the same priority as this job.

EXECUTING TASK STATUS

Q returns executing task status as a table with the following column headings:

<u>Heading</u>	<u>Description</u>
JDN	Job descriptor number associated with the job.
JOBNAME	For batch jobs, the eight-character batch input file name. For interactive sessions, the name of the highest level task.
USER	User number to which the task belongs (six decimal digits).
LID	Logical mainframe identifier of the remote system for which the job is destined (three characters).
PRI	Priority (two decimal digits) and subpriority (three decimal digits).
TASK	Name of the currently executing task.
STATUS	Task status identifier. The possible identifiers are listed in table 4-10.
TL	Available time remaining for the job or interactive session in system seconds (decimal integer).

<u>Heading</u>	<u>Description</u>
CBC	Number of 512-word blocks of memory currently assigned to the task (decimal integer).
CWS	Current working set size in 512-word blocks (decimal integer).
CLP	Current large page count (decimal integer).
NT	Current number of tape units reserved for the job or interactive session.

If you request executing task status but no tasks exist that can be listed for you, Q sends the following message to the job dayfile or interactive terminal.

NO JOBS EXECUTING

Table 4-10. Task Status Identifiers

Identifier	Meaning
SUSP RECALL	Suspended for an event or a time period.
RUNNING	Currently executing.
SUSP BY SYSTEM	Suspended for a system action.
SUSP BY OPERATOR	Suspended by operator.
WAIT SYS I/O	Suspended until I/O completes.
WAIT MF xxx	Suspended for communication with the specified system.
WAIT ALT/MEM	Suspended until the CPU is assigned to the task.

OUTPUT FILE STATUS

Q displays output file status as a table.

The line preceding the table column headings contains the following:

DEFAULT OUTPUT LID = lid

where lid identifies the remote host that is to receive queue files not explicitly designated to go elsewhere. Refer to the description of MFQUEUE or SUBMIT in this chapter or Queue File Transfers in chapter 3 for a more detailed discussion.

The output file status table has the following column headings:

<u>Heading</u>	<u>Description</u>
JDN	Job descriptor number.
JN/FN	Job name of batch output or file name of MFQUEUED file.
LID	The remote mainframe logical identifier.
ORIG	Original owner's user number.
SIZE	File length in 512-word blocks (decimal).
NAME	Logical file name of the output-file-family's last group file.
STATUS	A message indicating the status of the output-file-family. If normal status, this field is blank. The possible identifiers are listed in table 4-11.

Table 4-11. Output File Status Identifiers (Sheet 1 of 2)

Identifier	Meaning
DESTINATION LID DISABLED	The LID of the remote host to which the output is to be sent has been turned logically off.
DESTINATION NOT RESPONDING	The remote host to which the output is to be sent is not responding to connection requests.
DESTINATION REJECTING FILE	The remote host to which the output is being sent is not accepting the file. To obtain a more detailed explanation of the rejection, refer to chapter 2.
SIL ERR OCCURRED DURING FILE XFER	On the last attempt to send the output file to its destination, a SIL error occurred during I/O. To obtain a more detailed explanation of the error, refer to chapter 2.
DIVERTED	The operator used the DIVE command to divert the output to a different LID.
HARDWARE PATH TO LID NOT AVAILABLE	The NAD associated with the LID or path to the destination has been disabled.
SYS ERR OCCURRED DURING FILE XFER	On the last attempt to send the output file to its destination, an error occurred on a system call. To obtain a more detailed explanation of the error, refer to chapter 2.
UNDEFINED STATUS CODE xxx	xxx is the value of OSTAT, the output status code in the FILEI. This value is not defined; contact the site analyst.

Table 4-11. Output File Status Identifiers (Sheet 2 of 2)

Identifier	Meaning
RHF ERROR OCCURRED DURING FILE TRANSFER	On last attempt to send output file to its destination, an internal RHF error occurred during the transfer.
RWF ERROR OCCURRED DURING FILE TRANSFER	On last attempt to send output file to its destination, an internal RWF error occurred during the transfer.

If you request output file status but no files exist that can be listed, Q sends the following message to the job dayfile or interactive terminal.

NO JOBS IN OUTPUT QUEUE

REQUEST - CREATE LOCAL FILE

The REQUEST utility can create a local mass storage file, tape file, or a file connected to a terminal, depending on the value assigned to the DEVICE parameter.

REQUEST always creates a file index entry for the file, regardless of its device type.

The REQUEST control statement formats are shown in figure 4-49. Specify only those parameters valid for the device type.

The first parameter specified on the statement must be the file name. For mass storage files, the file length, if specified, must be the second parameter. All other parameters can be specified in any order.

REQUEST returns a message response after successful completion only if specified parameters are incompatible with the device type of file.

REQUEST cannot create a file having a security level higher than that of the job or interactive session.

The initial retention period for the file is defined by an installation parameter value. To specify a different retention period, execute the SWITCH utility.

The DEVICE=TE parameter option requesting a file connected to an interactive terminal can only be used interactively. If a batch job attempts to use it, REQUEST returns a fatal error.

Format for Mass Storage Files	
REQUEST, lfn/len, DEVICE=MS, ACCESS=acs, SECURITY=lv1, RLMIN=rlmin, RLMAX=rlmax, NOEXTEND, NOSEGMENT, PACK=packid, PC=pc, RMD=rmd, RT=rt, SFO=org, TYPE=type, AU=blocks.	
Format for Tape Files	
REQUEST, lfn, DEVICE=NT, DENSITY=den, TF=fmt, NEOI=neoi, NOS=noise, CM=cm, LT=ls1, VA=va, VSN=vsn-list, ACCESS=acs, IU=iu, RU=ru, RETRY=try, HEC=ec, RT=rt, BT=bt, RLMIN=rlmin, RLMAX=rlmax, PC=pc, RMD=rmd, RA=ra, MPRU=mpru, RPB=rpb, CONVERT=cvt, LPROC=lp, MESSAGE=msg.	
Format for Files Connected to a Terminal (valid for interactive use only)	
REQUEST, lfn, DEVICE=TE, ACCESS=acs, SECURITY=lv1.	
lfn	File name: one to eight letters or digits beginning with a letter. lfn is a required parameter and must be the first parameter specified.
<u>DEVICE</u> =dev	Device type on which file resides.
MS	Mass storage
NT	Magnetic tape
TE	Interactive terminal (valid only for interactive REQUEST).
If DEVICE=dev is omitted, DEVICE=MS is assumed.	

Figure 4-49. REQUEST Control Statement Format (Sheet 1 of 7)

<u>ACCESS=acs</u>	Access permission set (any combination of the following letters without separators).													
	<table> <tr> <td>R</td> <td>Read permission</td> <td></td> </tr> <tr> <td>W</td> <td>Write permission</td> <td></td> </tr> <tr> <td>X</td> <td>Execute permission</td> <td rowspan="3">} Applicable to mass storage files only</td> </tr> <tr> <td>A</td> <td>Append permission</td> </tr> <tr> <td>M</td> <td>Modify permission</td> </tr> </table>	R	Read permission		W	Write permission		X	Execute permission	} Applicable to mass storage files only	A	Append permission	M	Modify permission
R	Read permission													
W	Write permission													
X	Execute permission	} Applicable to mass storage files only												
A	Append permission													
M	Modify permission													
	If ACCESS=acs is omitted, ACCESS=RWXAM is assumed for mass storage files, ACCESS=R is assumed for tape files, and ACCESS=RW is assumed for files connected to a terminal.													
<u>SECURITY=lvl</u>	Security level (1 through 8). The specified security level cannot be greater than that of the job or interactive session. If SECURITY=lvl is omitted, the security level of the job or interactive session is used.													
<u>Parameters for Mass Storage Files Only</u>														
<u>len</u>	Number of 512-word blocks initially allocated for the file (decimal or hexadecimal number between 1 and 976563). If len is not specified, the default is 8 blocks.													
<u>NOEXTEND</u>	Inhibit automatic file extension by VSOS. The file cannot extend past the length specified. If NOEXTEND is omitted, extension is allowed.													
<u>NOSEGMENT</u>	Inhibit disk segmentation by VSOS. The file must be allocated as a contiguous area on disk. If NOSEGMENT is omitted, segmentation is allowed.													
<u>PACK=packid</u>	A disk pack in the device set on which the file resides. If PACK=packid is omitted, VSOS chooses the pack.													
<u>SFO=org</u>	File organization. <table> <tr> <td>S</td> <td>Sequential</td> </tr> <tr> <td>D</td> <td>Direct</td> </tr> </table> <p>If SFO=org is omitted, the installation-defined default file organization (released value, S) is used.</p>	S	Sequential	D	Direct									
S	Sequential													
D	Direct													
<u>TYPE=type</u>	File data type for VSOS. <table> <tr> <td>P</td> <td>Physical data</td> </tr> <tr> <td>C</td> <td>Virtual code</td> </tr> </table> <p>P is suitable for all files except controllees. C is suitable for controllees only. If TYPE=type is omitted, P is used.</p>	P	Physical data	C	Virtual code									
P	Physical data													
C	Virtual code													

Figure 4-49. REQUEST Control Statement Format (Sheet 2 of 7)

Parameters for Mass Storage Files Only

AU=blocks Allocation unit. Allows you to aid performance by giving the system a guideline on the integer number of 512-word blocks to allocate when the file is extended. The value range of blocks is 1 to 65,535. If the file is created and blocks is not a multiple of the DAU for the device where the first allocation occurs, blocks is rounded up to the next multiple of the DAU. If the file is already local, this parameter is ignored.

Parameters for Mass Storage and Tape Files Only

RLMIN=rlmin Minimum record length in bytes. If RLMIN=rlmin is omitted, the minimum record length is one byte.

RLMAX=rlmax Maximum record length in bytes, or for F format records, the fixed record length. If RLMAX=rlmax is omitted, the maximum record length is an installation parameter value (released value, zero).

PC=pc Padding character. It may be specified as a single character or as a number in the range of 0 - 255. If PC=pc is omitted, the installation-defined default character [released value, blank (code #20)] is used.

RMD=rmd Character used as the record delimiter for R format records. It may be specified as a single character or as a number in the range of 0 - 255. If RMD=rmd is omitted, the installation-defined default character [released value, US (code #1F)] is used.

RT=rt Record type.

B	System block (valid for tape files only)
F	ANSI fixed length
L	CYBER Record Manager control word (valid for tape files only)
R	Record mark delimited
U	Undefined
W	Control word delimited

If SFO=D is specified for a mass storage file, the only valid record format is F.

If RT=rt is omitted and the file is a sequential access mass storage file or tape file, the installation-defined default type (released value, R) is assumed. If RT=rt is omitted and the file is a direct access mass storage file, F format is assumed.

Figure 4-49. REQUEST Control Statement Format (Sheet 3 of 7)

Parameters for Tape Files Only

DENSITY=den Recording density.

 PE 1600 cpi

 GE 6250 cpi

If DENSITY=den is omitted, the installation-defined default density (released value, 6250 cpi) is used.

TF=fmt Tape data format.

 I NOS internal format

 SI System internal format

 LB Large block format

 V Variable block format

 NV Variable format tape with embedded tape marks

If TF=fmt is omitted, the installation-defined default format (released value, LB) is used.

NEOI=neoi No EOI detection. Valid only for TF=NV, LT=V tapes.

Y Does not return EOI status.

N Returns EOI status when two consecutive tape marks are encountered.

If NEOI=neoi is omitted, N is used.

NS=noise Noise size in bytes; this is used only while reading tapes generated on a non-CYBER 200 system. $0 \leq \text{noise} \leq 31$. Default value is zero.

CM=cm Character conversion mode. If the tape is labeled, the conversion mode must match the conversion mode used when the labels were written. Conversion is performed only if CONVERT is specified on the REQUEST or LABEL statement for the file.

 AS ASCII character set

 EB EBCDIC character set

If CM=cm is omitted and the tape is labeled, the default mode is the conversion mode used when the labels were written. If CM=cm is omitted and the tape is unlabeled, the installation-defined default mode (released value, ASCII) is used.

Figure 4-49. REQUEST Control Statement Format (Sheet 4 of 7)

Parameters for Tape Files Only

LT=lsl Indicates the types of labels.

AN	ANSI standard labels
NS	Nonstandard labels
U	Unlabeled file

If LT=lsl is omitted, the file is assumed to be an ANSI standard labeled file.

VA=va Volume accessibility character in the VOL1 label. This parameter is required if the character is nonblank.

VSN=vsu-list List of tape volumes associated with the file name (1 to 255 VSNs separated by commas; each VSN consists of one to six characters). If VSN=vsu-list is omitted, the operator assigns tape volumes to the file as needed.

IU=iu Inhibit unload option indicating whether the system unloads a tape volume when its file is returned.

Y	Does not unload tape volume
N	Unloads tape volume

If IU=iu is omitted, N is used.

RU=iu Read unconditional option indicating whether the system allows reads past the end-of-tape (EOT) reflective marker on the tape volume.

Y	Allows reads past EOT
N	Does not allow reads past EOT

If RU=ru is omitted, N is used.

RETRY=try Error retry option indicating whether, if a data error is detected, the system performs its standard error recovery procedures.

Y	Error recovery performed
N	Error recovery not performed

If RETRY=try is omitted, error recovery is performed.

Figure 4-49. REQUEST Control Statement Format (Sheet 5 of 7)

Parameters for Tape Files Only

HEC=hec Hardware error correction option for GCR tapes (6250 cpi). If enabled by an installation parameter, the option indicates whether the system allows the writing of single-track errors that can be corrected as the tape is read (on-the-fly correction).

Y Single-track errors allowed

N Standard error recovery performed for single-track errors

If HEC=hec is omitted, Y is used.

BT=bt Blocking type.

I Internal

C Character count

K Record count

If BT=bt is omitted, character count blocking is used.

RA=ra Ring access option. Not valid if ACCESS= is specified.

Y If a tape is already mounted with ring in, Read/Write access is assigned; otherwise, Read only access is assigned.

N Access defaults to Read only consecutive tape marks are encountered.

If RA=ra is omitted, N is used.

MPRU=mpru Maximum PRU size in bytes; used only if the file uses the V tape format. If MPRU=mpru is omitted, the MPRU size for V format is 32768 bytes.

RPB=rpb Records per block; used only for the K blocking type. If RPB=rpb is omitted, one record per block is assumed.

CONVERT=cvt Data conversion option. If CONVERT is omitted, no conversion is performed and the data is read and written as binary data.

The values for cvt are these:

Y Tape data is read and written as character codes, using the character set specified by the CM parameter.

N No conversion is performed.

If CONVERT= is specified as YES on the REQUEST control statement, setting either YES or NO on the LABEL control statement has no effect and conversion is done. If CONVERT= is not specified on REQUEST, LABEL can be set it to either YES or NO with the expected results.

Figure 4-49. REQUEST Control Statement Format (Sheet 6 of 7)

Parameters for Tape Files Only

LPROC=lp Label processing option.

 R Read existing labels (verify existing HDR1 label).

 W Write new labels.

If LPROC=lp is omitted, label processing depends on the value of the ACCESS parameter. For ACCESS=R or ACCESS=RW, LPROC=R is assumed. For ACCESS=W, LPROC=W is assumed.

MESSAGE=msg Message displayed on the operator's O display (1 to 64 characters). If MESSAGE is omitted, no message is displayed. The msg is enclosed by quotation marks. The message appears in the O display after the system mount message.

Figure 4-49. REQUEST Control Statement Format (Sheet 7 of 7)

FILE SPACE ALLOCATION

REQUEST determines whether a given pack has adequate space to hold a file of the required size; if not, the utility returns a fatal error code. This utility allocates mass storage for the file and controls whether the file space is contiguous at creation and whether the file can be extended.

The NOSEGMENT and NOEXTEND parameters determine how space is allocated for a mass storage file. For more information, refer to File Space Allocation in chapter 2 of this manual. The interaction between the NOSEGMENT and NOEXTEND parameters is as follows:

<u>Extendable File</u>	<u>Segmentable File</u>	<u>Result</u>
No	No	One segment. File cannot be extended.
No	Yes	File created as one or more segments. File cannot be extended.
Yes	No	File created as one segment. Noncontiguous segments cannot be added.
Yes	Yes	File created as one or more segments. Noncontiguous segments can be added.

TAPE FILE REQUEST

A REQUEST statement that specifies DEVICE=NT can request either a single tape file or a multifile set. The same parameters are valid for either a single file or multifile set.

NOTE

To request a tape file, reserve one or more tape drives on the RESOURCE statement of the job.

For an interactive job, the resource allocations for tape drives are done at request time. If all drives are committed, an error is returned and the request is terminated.

The REQUEST statement specifies the VSNs of the tape volumes associated with the file or multifile set. If a tape volume is not specified, the operator assigns tape volumes to the job as needed.

Tape Labels

The REQUEST statement specifies whether the file is unlabeled or labeled and, if labeled, whether it is an ANSI standard labeled file or nonstandard labeled file.

NOTE

Nonstandard labels cannot be read or written unless the site changes the released value of the IP_TPNSL installation parameter.

If the REQUEST statement specifies ANSI standard labels and the ANSI standard label VOL1 has already been written on the first tape volume (the tape is blank labeled), the label determines the recording density and conversion mode for the volume.

If the accessibility character in the VOL1 labels of the tape volumes is nonblank, specify the accessibility character on the VA parameter.

Data Format Specification

Only the tape data format can be specified on the REQUEST statement. However, other data format parameter values specified on the REQUEST statement can be overridden if you specify the same parameter on the LABEL statement for the file.

For a multifile set, the REQUEST parameter values apply to all files in the set. If the LABEL statement for a file in the set overrides a REQUEST data format parameter value, the changed value applies only to that file.

Processing Options

The following tape processing options can be specified only on the REQUEST statement.

- IU: Inhibit unload option indicating whether a tape volume is unloaded when its file is returned.
- RU: Read unconditional option. If the option is allowed by an installation parameter, this option can enable reading past the end-of-tape reflective marker.
- RETRY: Error retry option indicating whether the standard error recovery procedures are used.
- HEC: Hardware error correction option for GCR tapes (6250 cpi). If enabled by an installation parameter, specify that the system is to allow the writing of single-track errors that can be corrected as the tape is read (on-the-fly correction).
- NEOI: No EOI detection option for NV format tapes indicating whether an EOI status is returned when two consecutive tape marks are encountered.

Operator Message

A message to the operator can be specified on the REQUEST statement. It appears on the operator's O display after the prompt requesting that the operator mount the tape.

For example, suppose the job contains the following statement:

```
REQUEST,X,DEV=NT,VSN=MYVSN,MESSAGE="TAPE IS  
IN I/O BOX ARHN".
```

The following prompt appears on the O display.

```
MOUNT VSN=MYVSN DENSITY=GE RING=OUT  
TAPE IS IN I/O BOX ARHN
```

A message can also direct the operator to mount an unlabeled tape. For example, suppose a job contains the following statement:

```
REQUEST,X,DEV=NT,MESSAGE="MOUNT SCRATCH TAPE".
```

The following prompt appears on the O display.

```
NO INITIAL VSN SPECIFIED. TYPE j.VSN,vsn  
MOUNT SCRATCH TAPE
```

RERUN - SET RERUN STATUS

The RERUN control statement reverses the effects of a preceding NORERUN control statement. The statement is valid only within a batch job.

If rerun status is set for a job, the batch input file can be rerun from its beginning (at whatever priority it was running) when VSOS is reautoloaded after a system failure that terminated the batch processor.

An installation parameter can prevent the rerunning of jobs.

RERUN format is shown in figure 4-50. Figure 4-35 shows RERUN use.

```
RERUN.
```

Figure 4-50. RERUN Control Statement Format

RESOURCE - SET JOB RESOURCE LIMITS

The RESOURCE control statement (refer to figure 4-51) can establish the time limit, job category, priority, working set size limit, and large page limit for a batch job. It can also reserve tape drives for use by the job.

The RESOURCE statement is only valid in a batch job.

The RESOURCE statement is optional; each of its parameters is also optional. If the statement or any of its parameters are omitted, the system uses default values.

If specified, the RESOURCE statement must follow the USER statement in the job file. (The first statement in the file is called the job statement and is used by the software that received the file from the front-end system; it is then removed from the file. The second statement in the file is the USER statement. RHF uses the USER statement.)

The RESOURCE statement can specify the job category of the job. Job categories are described under Resource Allocation in chapter 3 of this manual. Except for the default job categories, JDEFAULT, for batch jobs and INTRACTV for interactive tasks, you must ask installation personnel for the job categories for which you are validated.

The RESOURCE statement can specify a job priority. The effect of the priority specification is described under Job Scheduling in chapter 3 of this manual.

RESOURCE, TL=t, JCAT=j, PRIORITY=p, WS=w, LP=lp, NT=nt.

TL=t Job time limit in system seconds† (decimal integer between 1 and 599940).

If TL=t is omitted, the job time limit is the default for the job category selected.

JCAT=j Job category name (one to eight letters or digits). (Site personnel define the job category names and limits.)

NOTE

Do not specify a job category whose maximum working set size is less than the maximum working set size required by the job. If the job requires all allocatable memory (a machine-size job), the maximum working set size of its job category must be all allocatable memory.

†A system second is one million STUs. If desired, an installation can substitute SBUs for system seconds as the time limit unit. The calculation of an STU or an SBU is described in volume 2 of this manual.

Figure 4-51. RESOURCE Control Statement Format (Sheet 1 of 2)

PRIORITY=p

Job priority, 1 (lowest) to 15 (highest). If the specified priority exceeds the maximum priority for the job category, the job priority is set at the maximum for its category.

If PRIORITY=p is omitted, the job priority is the installation-specified default priority for its category.

WS=w

Maximum working set size in blocks (decimal integer).

If WS=w is omitted, the maximum working set size for the job is the maximum working set size for its job category. Specifying WS=* notifies the input queue manager that the job requires all allocatable memory (a machine-size job).

NOTE

Use the WS parameter only for the following jobs:

- A machine-size job requiring all allocatable memory (specify WS=*)
- A job known to execute efficiently with a maximum working set size less than the maximum working set size of its job category

Misuse of the parameter could result in suspension of the job to prevent system performance degradation. The system automatically resumes the job when system resources are available.

LP=lp

Maximum number of large pages necessary for any task in the job (decimal integer).

If the large page limit, when multiplied by 128, exceeds the working set size limit, the job is aborted.

If any task in the job exceeds the limit, the job is aborted.

If LP=lp is omitted, the large page limit is set by an installation parameter (released value, 0).

NT=nt

Maximum number of tape drives needed by this job at any one time. This parameter is required if the job uses tape files. If NT=nt is omitted, the default value is 0.

Figure 4-51. RESOURCE Control Statement Format (Sheet 2 of 2)

TAPE DRIVE RESERVATION

If a job uses tape files, specify the NT parameter on its RESOURCE statement. The NT parameter specifies the maximum number of tape drives needed by the job at one time.

The input queue manager holds the job in the input queue until the number of drives requested is available. If the number requested is greater than the number of tape drives configured in the system, the job is held in the input queue until the operator evicts it.

The requested number of drives are reserved for the job during its execution unless a RETURN statement decrements the reserve count when it returns a tape file.

RETURN - EVICT LOCAL FILES OR DETACH PERMANENT FILES

The RETURN control statement (refer to figure 4-52) can perform any one of the following functions:

- Release mass storage space allocated to one or more local files
- Detach one or more permanent files from the job
- Release reserved tape drives

```
RETURN, { lfn-list  
        * }, DRC=drc.
```

lfn-list List of 1 through 16 private file names. Each name must consist of from one to eight letters and digits, beginning with a letter (except drop file names).

***** Indicates that the system should return all files attached or assigned to the job or interactive session, including files attached but not owned by the user.

DRC=drc Decrement reservation count option. This determines whether RETURN decrements the number of tape drives reserved for the job when it returns a tape file.

YES Decrement the number of reserved drives.

NO Do not decrement the number of reserved drives.

If DRC=drc is omitted, DRC=NO is assumed.

Figure 4-52. RETURN Control Statement Format

Files cannot be returned while they are open to a task.

Only files assigned or attached to this job can be returned.

If RETURN cannot return a specified file, it sends an error message to the job dayfile or to the interactive terminal.

When the RETURN statement specifies the * parameter, all files are returned except the following:

- Batch input file opened by the batch processor
- The job dayfile (file Q5DAYFLE)
- File Q5JOBFLE containing the control statement group of the executing batch job
- The drop file for the RETURN utility
- File Q5INPUT containing the input for the next task of the executing batch job

If RETURN returns files required by an executing job, the job can terminate abnormally.

RETURNING TAPE FILES

When RETURN returns a multifile set, it returns all files in the multifile set.

When a tape file is returned, RETURN checks whether the DRC parameter is specified. If DRC=YES is specified, it decrements the number of tape drives reserved for the job. The RESOURCE statement for the job specifies the initial tape drive reservation number. For a labeled tape, the DRC option is ignored if a local file (of a multifile set) is returned.

For an interactive user, the DRC parameter is ignored. The system always decrements the number of reserved drives.

REWIND - REWIND A TAPE FILE

The REWIND control statement rewinds one or more tape files.

A rewound tape file is positioned at the beginning of information (BOI). The BOI of an unlabeled tape file is the load point of the current tape volume. The BOI of a labeled tape file is immediately after its HDR1 label. If the file is a multivolume file and its HDR1 label is on a previous volume, the file is reassigned to the previous volume, which is positioned after the HDR1 label of the file.

REWIND attempts to rewind each specified file. It returns a message for each file it cannot rewind.

If more than one file on a multifile set is specified, the tape is positioned at the beginning of the last lfn in the list at the completion of the statement.

If a file that is not a tape file on a REWIND statement is specified, no action is taken for the file.

Figure 4-53 shows the REWIND control statement format.

REWIND, lfn-list.

lfn-list List of one or more file names separated by commas or blanks. This parameter is required.

Figure 4-53. REWIND Control Statement Format

SET - CHANGE JOB CHARACTERISTICS

The SET control statement (refer to figure 4-54) can change the current memory limits (the working set size limit or the large page limit) and the automatic dayfile routing for the job.

The current large page limit cannot exceed the current working set limit.

The initial memory limits for the job are specified on the job RESOURCE statement or by default values.

SET, WS=w, LP=lp, DAYFILE= $\left\{ \begin{array}{l} \text{ON} \\ \text{OFF} \end{array} \right\}$, DUMP=dp, TASKDUMP=dp, TASTL=t, TL=t, Rn=op value.

WS=w Current working set size limit in blocks (decimal integer). If the specified limit exceeds the maximum working set size limit for the job category or if the specified limit is smaller than the current large page limit (multiplied by 128), the job is terminated.

If WS=w is omitted, the current working set size limit is not changed.

LP=lp Current large page limit (decimal integer). If the specified limit exceeds the maximum large page limit for the job or the current working set limit, the job is terminated.

If LP=lp is omitted, the current large page limit is not changed.

DAYFILE= $\left\{ \begin{array}{l} \text{ON} \\ \text{OFF} \end{array} \right\}$ Specifies the current state of job dayfile routing. The job dayfile routing state at job termination determines whether the dayfile is routed or suppressed. If the state is ON, the dayfile is routed after the output files. If the state is OFF, the output files are routed but the dayfile is suppressed. Instead of the dayfile, the following message is sent.

SET(DAYFILE=OFF) USER DAYFILE SUPPRESSED

The job dayfile routing state can be changed at any time within the job, but only the state at job termination affects dayfile processing. The initial dayfile routing state is ON.

Figure 4-54. SET Control Statement Format (Sheet 1 of 2)

<u>DUMP</u> =dp†	dp=0	Turns off dump processing for the job.
	dp=controllee	Substitutes another dump processor for DUMP during the job.
<u>TASKDUMP</u> =dp†	dp=0	Turns off dump processing for only the next task in the job.
	dp=controllee	Substitutes a dump processor only for the next task in the job.
<u>TASKTL</u> =t		Sets the maximum time allowed for the next task. The actual time given for the next task will be no greater than the remaining time for the job.
<u>TL</u> =t		Changes the job's remaining time limit from its current value to the specified value. This stays in effect until the next SET,TL= or end of job.
		t is a decimal integer between 1 and 599940, specifying seconds.
Rn=op value		Sets or alters the control statement variable Rn.
	Rn	Identifies a global control statement variable. n is a number from 0 through 9.
	op	Defines the operation to be performed on Rn. If omitted, the value is placed in Rn. If positive, the value is added to the current Rn value. If negative, the value is subtracted from the current Rn value.
	value	Defines the value to be placed in, added to, or subtracted from Rn. The value may be one of the following:
	m	A decimal value. If preceded by #, the value is hexadecimal.
	Rn	A global control statement variable.
	RC	The last return code returned by a controllee.
	TV	The current threshold value.
	"string"	A character string (eight characters) maximum, preceded and followed by ". The string may not include ". If the string is less than eight characters, it is left-justified and blank-filled.
		The new value assigned to Rn is echoed back to the dayfile with the following message:
		Rn=#value_in_hex

Figure 4-54. SET Control Statement Format (Sheet 2 of 2)

†Both DUMP and TASKDUMP can be specified, with TASKDUMP taking precedence.

SKIP - REPOSITION A TAPE FILE

The SKIP control statement (figure 4-55) repositions a tape file forward or backward one or more partitions. The partition used can be logical record unit (L), tape group (G), or file (F).

If, while attempting to reposition the file, SKIP encounters a partition delimiter at a higher level than the specified partition, it stops repositioning the file. If it was skipping forward, the file is left positioned before the partition delimiter. If it was skipping backward, the file is left positioned after the partition delimiter.

If the specified file is not a tape file, no action is taken.

SKIP, lfn, PARTITION=part, NUMBER=n.	
lfn	Name of tape file to be repositioned.
<u>PARTITION</u> =part	Partition type.
	G Tape group
	F File
	L LRU
	If PARTITION=part is omitted, L is used.
<u>NUMBER</u> =n	Number and direction of partitions to skip. The direction is indicated by the sign of the number; a negative number causes a backward skip. If NUMBER is omitted, 1 is used.

Figure 4-55. SKIP Control Statement Format

SLGEN - CONSTRUCT SYSTEM SHARED LIBRARY

This control statement constructs the system shared library file. If you use SLGEN, you must supply the SYSLIB and the binaries of any utilities that are to be placed on the system shared library file.

The recommended method of building a new system shared library is as follows:

- Use OLE to build a dynamic library containing SYSLIB. Let ORIGIN default.
- Use SLGEN to build the new system shared library specifying the dynamic library as the old library used as input. Let ORIGIN default.

This method enables the various modules and utilities included in the new system shared library to be loaded more compactly than otherwise. This has a beneficial effect on paging during operation of the system using the system shared library.

The control statement format is shown in figure 4-56.

SLGEN,oldlib,LIBRARY=libfile,VERSION=version,LIST=listfile/len,INPUT=infile,ORIGIN=bitaddr.

<u>oldlib</u>	If specified, SLGEN updates the specified old attached system shared library file, oldlibs, based on the directives specified by the INPUT parameter, and places the results on the file specified by the LIBRARY parameter. If not specified, SLGEN constructs a new system shared library file, based on the directives specified by the INPUT parameter, and places the results on the file specified by the LIBRARY parameter. The file specified on the LIBRARY parameter may be the same as oldlib. When oldlib is specified, utilities being replaced are always put at the end of the library file. This causes the system shared library to grow in length. A new library must be built without an oldlib to get a compact system shared library.
<u>LIBRARY=libfile</u>	SLGEN places the new system shared library on this file. libfile is the name of the shared library. The default is NEWSLIB. SLGEN creates libfile to the exact size needed. libfile is always created by SLGEN as a local file.
<u>VERSION=version</u>	version is the user-defined version (up to eight characters in length) to be placed in the system shared library header. Default is blanks.
<u>LIST=listfile</u>	listfile is a local or attached file to which SLGEN writes the system shared library map. If omitted, the map is produced on the file OUTPUT. If LIST=0, no map is produced.
<u>INPUT=infile</u>	infile is a local or attached file from which SLGEN reads the directives (listed in figure 4-57). The default is INPUT. If there are no directives, the new system shared library is equivalent to the old one. The directive file is terminated by an EOF or the END directive.

Figure 4-56. SLGEN Control Statement Format (Sheet 1 of 2)

<u>ORIGIN</u> =bitaddr	Starting bit address for the shared library. ORIGIN defaults to #800000000000 and, if specified, it must be less than #7FFF00000000. ORIGIN is used to construct a shared library that can be used and debugged in the user's address space. ORIGIN cannot be used if the oldlib parameter is specified.
------------------------	--

Figure 4-56. SLGEN Control Statement Format (Sheet 2 of 2)

The SLGEN directive formats are shown in figure 4-57.

LIB,dirfile. dirfile is the name of an attached or local OLE generated library file that contains the SYSLIB modules that are to be placed in the new system shared library file. If LIB is not specified, SLGEN retains the SYSLIB that is in the old system shared library file. If LIB is specified, the SYSLIB in the old system shared library file is deactivated and the new one is added. Only one LIB directive may be specified in the directive file. The new system shared library file must contain an active SYSLIB. This directive must be specified before any UTL directive that expects to use the SYSLIB.

UTL,utname,loader options.

<u>Parameter</u>	<u>Description</u>
utname	utname is the name of a utility that is to be added or replaced on the new system shared library file. utname is required. If utname exists on the old system shared library, it is deactivated and the new utility is added. The new system shared library file must contain the linker utility. The linker utility must be added before SYSLIB is added and before any other utilities. When a utility is replaced, the code for the old copy still exists in the system shared library, and any copies of the old controllee file that were statically linked to the shared library will still execute.
loader options	This parameter tells how to load the utility. The only loader options that are required are the list of object code files (they must appear first) and the CNTROLEE parameter. The syntax and rules of the option are the same as those on the LOAD control statement. The option ORIGIN cannot be specified as it is supplied by SLGEN. No grouping of code blocks or originating of code blocks is allowed. The defaults for unspecified options are the same as those specified by the LOAD control statement. It is your responsibility to ensure that there are no output file name conflicts between directive loader options.

END. Indicates that there are no more directives.

Figure 4-57. SLGEN Directive Formats

An example of a directive file for building a library follows.

```
a) UTL, linker, BLINKER,EN=Q9LINK,TSP=1,LINK=M.  
LIB, SYSLIB.  
UTL, utility that needs SYSLIB  
UTL, utility that needs SYSLIB  
.  
.  
.  
END.  
  
b) UTL, linker,BLINKER,EN=Q9LINK,TSP=1,LINK=M.  
UTL, utility that does not need SYSLIB  
.  
.  
.  
LIB, SYSLIB.  
UTL, utility that needs SYSLIB, ... .  
.  
.  
.  
END.
```

SUBMIT - SUBMIT A FILE TO A QUEUE

SUBMIT allows you (whether as an interactive or a batch user) to submit a file to the local CYBER 200 input queue directly and route output as desired. The submitted job may specify the user number currently in use or some other user number. Once the job enters the input queue, its subsequent processing conforms with the processing of any other job in the system.

The file being submitted must begin with a job card followed by a valid user control statement. The job card must contain 1 through 8 alphanumeric characters, the first being alphabetic, and must be terminated by a valid job control statement terminator.

The file that is submitted to the input queue is a copy of the file with last group characters and routing information attached. Your file remains unmodified with the job/session.

The format of a SUBMIT control statement is shown in figure 4-58.

SUBMIT,lfn,ST=lid,JN=yyy, { INPUT=filename, JCS="string1","string2",...,"stringn" }.	
lfn	The logical name of the file being submitted. This file must be either a local file or an attached permanent file. This parameter is required.
ST=lid	The RHF logical identifier of the remote system (three ASCII alphanumeric characters) to which output is sent. The identifier is checked for validity as a logical identifier and is matched against the set of available logical identifiers. This parameter is optional. If ST is not specified and if SUBMIT is executing from within a batch job, the output is sent to the remote system that submitted the original batch job. If ST is not specified and if the SUBMIT is performed interactively, the output is sent to the remote system specified by the default output LID listed in the H,0 display or the Q,0 display.
JN=yyy	Specifies the job name of the submitted job. This parameter is optional. If JN = yyy is omitted, the job card name is used. yyy must be a one- to eight-character sequence that is a valid file name.

Figure 4-58. SUBMIT Control Statement Format (Sheet 1 of 2)

INPUT=filename

If INPUT=filename is specified, text strings are read from filename. Each text string is interpreted as a line of text by the front end and is subject to the restrictions of that machine.

If the keyword INPUT is used in stand-alone fashion (using the second-level default), text strings are read from file INPUT.

If this parameter is not specified, text strings from the JCS parameter are used; if neither JCS nor INPUT is specified, no explicit text is sent to the other system. For a batch job SUBMIT, if neither JCS nor INPUT is specified, the implicit text of the batch job will be used to determine output routing.

The JCS and INPUT parameters cannot both be specified in one control statement.

JCS="string1",
"string2",...,
"stringn"

If JCS= is specified, these text strings are control cards that are to be executed as explicit routing text when the submitted job has terminated. Each text string is interpreted as a line of text and executed by the front end (ST=lid), and each is subject to the restrictions of that machine.

The INPUT parameter may not be specified when this parameter is used.

Figure 4-58. SUBMIT Control Statement Format (Sheet 2 of 2)

A successful completion of SUBMIT results in the following message:

JOB SUBMITTED. JOBNAME=jobname,JDN=jdn.

SUMMARY - PROVIDE RESOURCE USAGE INFORMATION

The SUMMARY control statement provides information on cumulative resource usage of a batch job. The information is sent to the job dayfile.

The SUMMARY statement is valid only within batch jobs. It is executed by the batch processor itself.

Figure 4-59 shows the SUMMARY statement format. It has no parameters.

```
SUMMARY.
```

Figure 4-59. SUMMARY Control Statement Format

SUMMARY OUTPUT

The following are the messages SUMMARY can send to the job dayfile. SUMMARY sends a message only if the resource usage described by the message is greater than zero. Each of the following messages is preceded by the time at which the message was sent.

<u>Message</u>	<u>Description</u>
SUMMARY.	Initial message always sent.
CHARGE,account,project number	Number of SBUs/STUs used for the account and project number listed.
SYSTEM TIME UNITS (STU)	Number of STUs used (real value). The algorithm used to compute STUs is described in volume 2 of this manual.
SYSTEM BILLING UNITS (SBU)	Number of SBUs used (real value.) The algorithm used to compute SBUs is described in volume 2 of this manual.
USER CPU TIME (SECS)	Number of seconds of CPU time used. This does not include the CPU time used for virtual system requests.
SYSTEM CPU TIME (SECS)	Number of seconds of CPU time used for virtual system requests.
USER MEMORY USAGE BLOCKS * [BLOCKS * (SECS)]	Memory used as computed by multiplying the number of blocks in the current working set by the number of CPU seconds used.
USER AVERAGE WORKING SET SIZE (BLOCKS)	Total memory usage in blocks divided by the user CPU time value.
NUMBER OF VIRTUAL SYSTEM REQUESTS	Number of calls to virtual system routines.
NUMBER OF SMALL PAGE FAULTS	Total number of small page requests minus the small page requests resulting from task swapping.

<u>Message</u>	<u>Description</u>
NUMBER OF LARGE PAGE FAULTS	Total number of large page requests minus the large page requests resulting from task swapping.
NUMBER OF DISK I/O REQUESTS	Number of explicit read and write requests and implicit write requests to mass storage.
NUMBER OF DISK SECTORS TRANSFERRED	Number of disk sectors transferred by explicit read and write requests and implicit write requests.

SWITCH - CHANGE FILE ATTRIBUTES

The SWITCH utility can change the attributes of a local file, an attached private file, or a public file. The changes made in file attributes are entered in the permanent file index entry for the file and therefore are kept until they are changed again or until the file is purged.

Only the file owner can change private file attributes. Only privileged users can change public file attributes.

Pool file attributes cannot be changed. However, the pool boss can use a GIVE statement to change a pool file to a private file, then use a SWITCH statement to change the private file attributes, and finally use another GIVE statement to give the private file to the pool.

SWITCH can change the following attributes of files residing on mass storage. The significance of the file structure attributes is described under Logical Record Format in chapter 2 of this manual.

- File name
- Account identifier
- Master project number
- File extension size
- Maximum or minimum record length
- Record type
- Padding character
- Record mark character
- File organization
- File type (controllee or data)
- File retention period
- Internal characteristics

File attributes cannot be changed while another user has the file attached or while a privileged user has the file open.

The SWITCH control statement format is shown in figure 4-60. The current name of the file must be the first parameter specified. If specified, the new name for the file must be the second parameter specified.

All other parameters are optional and can appear in any order.

SWITCH,olfn,nlfn,ACCOUNT=acct,MPN=mpn,IC=ic,RLMIN=rlmin,RLMAX=rlmax,RT=rt,PC=pc,
 RMD=rmd,SFO=sfo,TYPE=type,AU=blocks,RETENTION=days,BT=bt,LPROC=lp,
 MPRU=mpru,RPB=rpb.

olfn	Current name of the file whose attributes are to be changed. This parameter is required.
nlfn	New name for the file (one through eight letters or digits, beginning with a letter).
<u>ACCOUNT</u> =acct	New account identifiers (one to eight characters) assigned to the specified file.
<u>MPN</u> =mpn	New master project number (one to three alphanumeric characters) assigned to the specified file.
<u>RLMIN</u> =rlmin	New minimum record length in bytes. Valid values are $1 \leq \text{rlmin} \leq 2^{32}-1$.
<u>RLMAX</u> =rlmax	New maximum or fixed record length in bytes. Valid values are $1 \leq \text{rlmax} \leq 2^{32}-1$.
<u>RT</u> =rt	New record type.
	F ANSI fixed length
	R Record mark delimited
	U Undefined
	W Control word delimited
	B System block
	L Lower CYBER Record Manager control word
<u>PC</u> =pc	New ASCII padding character. Any ASCII character is valid.
<u>RMD</u> =rmd	New ASCII record mark character. Any ASCII character is valid.
<u>SFO</u> =sfo	New SIL file organization. If SFO=D is specified, the record format must be F and the file must be a mass storage file.
	D Direct access
	S Sequential access

Figure 4-60. SWITCH Control Statement Format (Sheet 1 of 2)

For Mass Storage Files Only

TYPE=type New file type.

 C Virtual code file (executable)

 P Physical data file (not executable)

AU=blocks Allocation unit. The number of 512-word blocks to be allocated when the file is extended. The value range of blocks is 1 to 65,535. If blocks is not a multiple of DAU for the device on which the segment is allocated, the system rounds it upward to the next multiple of the DAU.

RETENTION=days Number of days the file is to be retained on mass storage (0 through 1023).

IC=ic New file format.

 AS 8-bit ASCII code; ANSI carriage control if the file is a print file

 BI Binary

 PA 8-bit ASCII code; ASCII carriage control if the file is a print file

For Tape Files Only

BT=bt New blocking type.

 I Internal

 C Character count

 K Record count

If BT=bt is omitted, the blocking type is not changed.

LPROC=lp New label processing option.

 R Read existing labels (verify existing HDR1 label)

 W Write new labels

If LPROC=lp is omitted, the label processing option is not changed.

MPRU=mpru New MPRU size in bytes; used only if the file uses the V tape format. If MPRU=mpru is omitted, the MPRU size is not changed.

RPB=rpb New records per block; used only for the K blocking type. If RPB=rpb is omitted, the records per block is not changed.

Figure 4-60. SWITCH Control Statement Format (Sheet 2 of 2)

TASKATT - ALTER A TASK'S ATTRIBUTES

TASKATT is a utility that is used to alter a task's attributes that exist in the controllee's minus page. TASKATT uses the hierarchical search to find the file and must have write permission to alter the file.

The TASKATT control statement format is shown in figure 4-61.

TASKATT, filename, DFL=nn, SAVE=x, VALIDATE=N/Y, ULIB=1fn, SLIB=1fn, MAP=N/Y.

<u>Parameter</u>	<u>Description</u>
filename	Name of attached or local controllee file. Required parameter.
<u>DFL</u> =nn	nn is the controllee new drop file length in blocks. If the parameter is not specified, the length is not changed.
<u>SAVE</u> =x	x is either Y or N. Y indicates that the dropfile is to be saved on termination of this controllee. N indicates that drop files are saved only on controllee aborts. If the parameter is not specified, the saving or retaining of drop files is left as is.
<u>VALIDATE</u> =N/Y	Y indicates the system checks every time this task runs to make sure the task is using the same user dynamic and system shared library with which it was loaded. N indicates no validation is done. Y causes a warning message to be issued when the task is run using libraries with which it was not loaded.
<u>ULIB</u> =filename	ULIB is used to change the user dynamic library with which this controllee runs. filename is the name of the user's dynamic library file. TASKATT must have read access for this file.
<u>SLIB</u> =filename	SLIB is used to change the system shared library with which this controllee runs. filename is the name of a system shared library file. TASKATT must have read access for this file.
<u>MAP</u> =N/Y	Y indicates the drop file is to be mapped in by the linker during dynamic execution. This helps prevent the drop file map getting full, but does slow the execution. N is the default and causes the linker to fault for free space with no prior system map message issued.

Figure 4-61. TASKATT Control Statement Format

TV - SET THRESHOLD VALUE

The TV control statement (refer to figure 4-62) is valid only within a batch job. It can perform either of the following functions, depending on whether or not a + follows the specified value:

- Set a threshold value to be compared to the return codes only from succeeding job tasks (+ specified)
- Set a threshold value to be compared to the highest return code from preceding and succeeding job tasks (+ omitted)

TV,value+.

value Threshold value (0 through 255).

+ Indicates that the specified value should be compared against the return codes of succeeding job tasks. If + is omitted, the specified value is compared against the highest return code from preceding job tasks.

Figure 4-62. TV Control Statement Format

Each job task returns a code (its return code) to the batch processor upon completion of the task. The batch processor compares the return code to the current threshold value to determine whether job processing should continue. The released value is 4.

The return codes are compared to the current threshold value as follows:

- If the return code is less than or equal to the current threshold value, the job continues with the next job task.
- If the return code is greater than the current threshold value, the batch processor searches for the next EXIT statement in the job. If it finds an EXIT statement, the threshold value is set at 255 and job processing continues with the statement following the EXIT statement. If it does not find an EXIT statement, the job terminates immediately.

The threshold value is not checked if the system aborts a task. For more information, refer to Job Termination in chapter 3 of this manual.

With a TV control statement test, the return codes of preceding job steps can be tested independently from the batch processor test. If the + following the value specified on the TV statement is omitted, the batch processor compares the specified value with the highest return code returned by a preceding job task to determine whether job processing should continue.

If the highest return code is greater than the specified threshold value, the batch processor searches for an EXIT statement to terminate the job. If the highest return code is less than or equal to the specified value, the specified value becomes the new threshold value for the subsequent tasks in the job.

Utilities described in this manual return only codes ERROR and FATAL. Code ERROR corresponds to return code 4; code FATAL corresponds to return code 8.

The value a given task returns is established by a Q5TERM SIL call or a TERMINATE system message executed within the task.

USER - PROVIDE USER VALIDATION INFORMATION

The USER control statement, which always follows the job statement in a VSOS batch job or is the first statement in a set of PTFs file transfer directives (refer to figure 4-63), identifies the CYBER 200 user number to which the job belongs. Entry of the password for the user number validates its use.

The USER statement also specifies the account identifier to which job resource usage is charged and the security level of the job. A task within the job cannot access a file with a security level greater than the job security level.

USER,USER=userno,ACCOUNT=account,PASSWORD=password,SECURITY=n.

<u>USER</u> =userno	CYBER 200 user number (one to six decimal digits). This parameter is required.
<u>ACCOUNT</u> =account	CYBER 200 account identifier (one to eight ASCII characters). This parameter is optional. If ACCOUNT=account is omitted, the user's default account identifier is used.
<u>PASSWORD</u> =password	User password (one to eight ASCII characters). Site personnel determine during system installation whether user password entry is required or optional.
<u>SECURITY</u> =n	Security level for the job (1 to 8). If SECURITY=n is omitted, the security level is the default value chosen by the site.

Figure 4-63. USER Control Statement Format

UPDATE is a utility used to maintain and manipulate a mass storage file containing images of coded punch cards or their equivalent. The utility provides you with features that are a subset of the UPDATE capabilities available under the NOS or NOS/BE operating systems. The UPDATE card image file cannot be interchanged between these systems, however, since internal file structures differ between the operating systems.

Typical use of UPDATE involves maintenance of a group of FORTRAN subroutines or assembly language routines. For convenience, you may often specify each routine as a separate deck, so that one routine can be changed or extracted without affecting other routines in the file. Because each card image in the deck has its own UPDATE-supplied sequence number, each can be referenced individually. A card can be deleted and replaced by two others, for example, in order to correct a routine or to increase its functions. A deck can be extracted from the card image file in a format acceptable to a compiler or assembler and used as if it had been entered into the system as a punch deck. Once a source card is in the UPDATE card image file, any physical punch card can be destroyed.

A source deck that is to be maintained through UPDATE must be made a part of a special format file known as a program library. Creation of a program library is accomplished through UPDATE itself. Subsequently, the program library can be changed on an UPDATE correction run: new decks can be added, existing decks removed, or the contents of any deck changed.

The contents of a deck need only be images of coded cards. UPDATE makes no assumptions about card contents. While programs are customary contents, they are not required contents, and UPDATE is equally applicable to a set of data cards or any other text.

You control UPDATE operations through the parameters on the UPDATE control statement and through a file containing directives and text. The directives are supplementary instructions for UPDATE; the text consists of cards to be made part of the UPDATE card image file. Together, the directives and text are called the input stream.

EXAMPLES

An example of an UPDATE creation run in which several FORTRAN routines become a program library with three decks is shown in figure 5-1. The UPDATE control statement indicates that a new library is to be created, with the name MYDECKS. The input for UPDATE is specified as the file INPUT, which is also the default file name when the I parameter is omitted.

```
USER,U=012306,AC=ACCT1,PA=MINE.
RESOURCE,TL=5.
UPDATE,I=INPUT,N=MYDECKS.
DEFINE,MYDECKS.
7/8/9
*DECK MAIN
  PROGRAM MAIN . . .
  .
  .
  .
  END
*DECK SUBPROG
  SUBROUTINE SUB1 . . .
  .
  .
  .
  SUBROUTINE SUB2 . . .
  .
  .
  .
*DECK ERRPROG
  SUBROUTINE SUB3
  .
  .
  .
6/7/8/9
```

Figure 5-1. Typical UPDATE Creation Run

The first directive encountered in figure 5-1 is *DECK; therefore, UPDATE recognizes a creation run and begins construction of a new program library. All cards following *DECK, up until the second *DECK directive, are written as a deck with the name MAIN. The first card is assigned the identifier MAIN.2, the next MAIN.3, and so forth. (The *DECK directive itself is also part of the library and has the identifier MAIN.1.)

A new deck, with card identifiers in the form SUBPROG.n, begins when UPDATE encounters the second *DECK directive. In this example, the main program is in a deck with the same name as the program, two subroutines are in a deck with the name SUBPROG, and a third subroutine is in a deck with the name ERRPROG. At the end of the UPDATE run, a program library exists with three decks.

The DEFINE control statement makes the local file MYDECKS a permanent file. The file MYDECKS remains in the system after job termination.

Figure 5-2 shows a correction run using the program library created in figure 5-1. This example adds a new card with text DIMENSION UP(50) near the beginning of subroutine SUB2. The location of the insertion is identified by the card identifier assigned within the deck SUBPROG, not by the subroutine name.

```
USER,U=012306,AC=ACCT1,PA=MINE.  
RESOURCE,TL=5.  
UPDATE (Q,P=MYDECKS)  
FORTRAN (I=COMPILE)  
LOAD.  
GO.  
7/8/9  
*IDENT FIXIT  
*INSERT SUBPROG. 89  
    DIMENSION UP(50)  
*COMPILE SUBPROG,MAIN  
6/7/8/9
```

Figure 5-2. Typical UPDATE Correction Run

The UPDATE control statement in figure 5-2 identifies the existing program library with the P parameter. Since the Q parameter appears, it also instructs UPDATE to operate in quick mode rather than full mode. When UPDATE begins execution, it reads the next input group in the batch job, which is presumed to contain the input stream. The first card in this stream gives a name (FIXIT) to the corrections being made. The second card identifies the location at which the new card is to be added; namely, after the card with the identifier SUBPROG.89. Since the third card in the input stream does not correspond to a directive, it is considered a text card. Within the program library, it becomes identified as FIXIT.1. The last card in the input stream instructs UPDATE to write decks MAIN and SUBPROG to a file in a format suitable for input to the FORTRAN compiler. By default, in the absence of a C parameter on the UPDATE control statement, the file name is COMPILE.

The FORTRAN compiler call in figure 5-2 names a file COMPILE as having the program to compile. Output of compilation is then loaded and executed with the LOAD and GO control statements.

Neither the FORTRAN call nor execution is required in figure 5-2. They are shown only to provide the programmer with a source listing of active cards in the deck with their card identifiers or to confirm proper program execution.

GENERAL PROCESSING

During execution, UPDATE manipulates several files; they are known as the input file, new program library, source file, old program library, compile file, and listable output file. File operations depend on whether UPDATE is performing a creation run or a correction run.

An UPDATE run is defined as all operations having a program library that results from a single UPDATE call. Any given run is a creation run or a correction run. Creation and correction runs are described as follows:

- A creation run constructs a program library. It is the original transfer of punch cards or card images into UPDATE format.
- A correction run changes an existing program library. As a result of the run, a new program library might be generated; but the program library is new only in the sense that the changes are incorporated into the existing program library. All history information remains.

File names are specified by parameters of the UPDATE control statement, which are summarized in table 5-1. Table 5-2 shows a summary of UPDATE directives used during a run.

Table 5-1. Summary of UPDATE Call Parameters

Parameter	Function
C	Specify name of compile file.
D	Define compile file card image width, excluding UPDATE sequence information.
F	Select full update mode and source file and compile file contents.
G	Select recreation of specified idents and specify file name to receive recreated idents.
I	Specify name of file with input stream.
K	Determine the order of decks in the compile file.
L	Select listable output file contents.
N	Specify name of new program library file.
O	Specify name of listable output file; content is determined by L parameter.
P	Specify name of old program library file.
Q	Select quick update mode.
S	Specify name of source file; content includes common decks and is determined by mode.
T	Same as S, but omits common decks.
8	Define compile file card image width, including UPDATE sequence information.
*	Redefine master control character for directives.
/	Redefine control character for comments.

Table 5-2. Summary of UPDATE Directives (Sheet 1 of 2)

Directive Keyword Abbreviation	Directive Format	Use
*AF	*ADDFILE lfn, deck	Read creation directives and text from named file and insert after deck identified.
	*ADDFILE lfn,ident.seqnum	Read creation directives and text from named file and insert after card identified.
*B	*BEFORE ident,seqnum	Write subsequent text cards before card identified.
*CA	*CALL deck	Write common deck to compile file.
*CD	*COMDECK deck,NOPROP	Define common deck and propagation parameter.
*C	*COMPILE deck1,deck2,...,deckn	Write specified decks to compile file, source file, and new program library.
	*COMPILE deck1.deck2	Write inclusive range of decks to these files.
*DF	*DEFINE,name	Define name to be used in evaluating *IF conditions.
*DK	*DECK deck	Define deck to be included in program library.
*D	*DELETE ident1.seqnum,ident2.seqnum	Deactivate inclusive range of cards.
	*DELETE ident.seqnum	Deactivate specified card.
*EI	*ENDIF	End conditional text.
*ID	*IDENT idname,B=n,K=ident,U=ident	Define correction set, bias for seqnum, and whether specified correction sets must be known or unknown to process this set.
*IF	*IF type,name,num *IF -type,name,num	Conditionally write text following *IF to compile file.
*I	*INSERT ident.seqnum	Write subsequent text cards after card identified.
*M	*MOVE deck1,deck2	Move deck1 to follow deck2.
*PM	*PULLMOD ident1,ident2,...identn	Recreate specified ids from old program library.
	*PULLMOD ident1.ident2	Recreate inclusive range of from old program library.

Table 5-2. Summary of UPDATE Directives (Sheet 2 of 2)

Directive Keyword Abbreviation	Directive Format	Use
*PD	*PURDECK deck1,deck2,..., deckn	Permanently remove specified decks from program library.
	*PURDECK deck1.deck2	Permanently remove inclusive range of decks.
*P	*PURGE idname1,idname2,...,idname3	Permanently remove specified correction sets from program library.
	*PURGE idname1.idname2	Permanently remove inclusive range of correction sets.
	PURGE idname,	Permanently remove specified correction set and all sets introduced after it.
*RD	*READ lfn	Read directives and text from specified file.
*WI	*WIDTH datlen,idlen	Reset size of line image written to compile file.
*Y	*YANK idname1,idname2,...,idnamen	Temporarily remove specified correction sets from program library.
	*YANK idname1.idname2	Temporarily remove inclusive range of correction sets.
*YD	*YANKDECK deck1,deck2,...,deckn	Temporarily deactivate decks specified.
*/	*/ comment	Copy text to listable output file.

UPDATE MODE AND FILES

All files used by UPDATE must reside on mass storage; all files created by UPDATE reside on mass storage. Default length of all of these files is #100 512-word blocks. Any of the UPDATE default files are opened and used if they exist as attached permanent files. If the files do not already exist, UPDATE uses Q5GETFIL to create them as local files. An intermediate processing file created by UPDATE when a new program library is being created has the file name TEMNEWPL. Its default length is #400 512-word blocks or the length of the new program library, whichever is larger.

The files that UPDATE creates or uses are as follows:

- Input file
- New program library
- Source file
- Old program library
- Compile file
- List file
- Pullmod file

Each of these files has a default file name, but any other name can be specified through the appropriate parameter on the UPDATE control statement.

With one exception, all files used must be separate and distinct files: that is, using the same file name for both list and compile causes an execution-time error. However, either source or compile can be directed to the file OUTPUT if no list is desired (L=0) or the list output is directed to another file. The one exception is that the source file and the pullmod file can be the same file.

The content of any compile file, source file, or new program library produced during a correction run is affected by the UPDATE mode. The mode of an UPDATE run is determined by a combination of the omission or specification of the F and Q parameters on the UPDATE control statement.

Select normal (selective), full, or quick UPDATE mode by the following:

<u>Parameter</u>	<u>Mode</u>
Both F and Q omitted	Normal selective mode in which the only decks processed are those modified or otherwise selected for processing
F specified	Full mode in which all decks on the old program library are processed
Q specified	Quick mode in which only decks specified on COMPILE or ADDFILE directives are processed
Both F and Q specified	Quick mode

Input File

The input file contains the input stream. The input stream consists of directives that provide the details of UPDATE processing and any new cards to be added to the program library. The file name is specified by the I parameter of the UPDATE call; the default file name is INPUT. Input file records cannot be longer than 256 characters.

New Program Library

The new program library is the file of card images and internal information in a special format that only UPDATE can process. It contains a deck list of the names of all decks in the file and a directory of all correction sets introduced into the file. Each card is represented in a format that adds a card identifier and adds history and status information known as correction history bytes. Blanks are compressed out of the card image.

A new program library is an output file created by UPDATE. Initially, it is generated on a creation run. For subsequent correction runs, the previous new program library is used as an input file and identified as the old program library; a new program library that incorporates the changes made during a correction run is then output from the correction run. The file name is specified by the N parameter of the UPDATE call; the default file name is NEWPL.

Normally the new program library contains all of the information on the old program library as well as any new cards added during the run creating the new program library. The only operations that remove card images are PURGE, PURDECK directives, or an editing update run. (PURGE and PURDECK are described in this chapter). During an editing update run, only active cards are written to the new program library. Inactive cards are discarded. No sequence numbers are changed. Idents from which cards have been permanently removed by editing are marked on the list output file, on the new program library, and on all future new program libraries.

Source File

The source file is an output file during a correction run. It consists of card images that allow regeneration of a new program library in resequenced format during a subsequent creation run. Only active cards and decks are part of the source file. The file name is specified by the S parameter of the UPDATE call; the default file name is SOURCE. The content of the file is controlled by the T, F, and Q parameters. You are responsible for routing the file to a punch or other output device.

Old Program Library

The old program library is the file generated as a new program library in a previous creation or correction run. It contains a record of changes made since the program library was created. It is required for any correction run. The file name is specified by the P parameter of the UPDATE call; the default file name is OLDPL. The old program library is an R format file with blank compression.

Compile File

The compile file is an output file that contains a copy of a deck in the program library restored to a format that can be processed by a compiler or assembler. Only active cards in the deck are part of the compile file. The file name is specified by the C parameter of the UPDATE call; the default file name is COMPFILE. The content of the file is controlled by the directives and the F, Q, or K parameters of the UPDATE call, with the D or 8 parameters or the *WIDTH directive selecting the number of columns in the image of each card. The compile file is an R format file with blank compression.

You can generate a compile file without sequence numbers by selecting both the D and 8 options.

List File

The listable output file is the print file. It shows the card identifiers assigned by UPDATE. The programmer must use them to reference a card image in any future correction run. The file name is specified by the O parameter of the UPDATE call; the default file name is OUTPUT. Content of the file is controlled by the L parameter, with options that can select a listing of directives processed, errors, comments, and a list of card images in the program library.

Pullmod File

The pullmod file can be output during a correction run. It contains card images of recreated idents generated by *PULLMOD directives. The pullmod file has the same format as an input file and can be used as input to a subsequent UPDATE execution. The file name is specified by the G parameter of the UPDATE call. By default, the file name specified by the S parameter of the UPDATE call will be used if the S parameter is present. If not, the default file name is SOURCE.

CREATION OF PROGRAM LIBRARY

A creation run exists when the first directive of the input file, other than a comment, is DECK or COMDECK. If the first directive is READ and the first directive of the file being read is DECK or COMDECK, a creation run also exists. Even if an old program library file is assigned to the job, UPDATE ignores its existence and processes the run in creation mode.

Directives that can be used in a creation run are limited to the following:

- READ
- DECK
- COMDECK

In a creation run, each DECK or COMDECK directive defines a deck to be inserted into the program library under construction. UPDATE decks can be one of two types: regular decks or common decks. They differ in that common decks can be called by name so that they are inserted into the text of another deck when the compile file is being generated. One copy of the common deck exists on storage, but multiple copies can be part of an output file.

In practice, the text written to a program library is often FORTRAN or assembly language routines in punch card format. UPDATE considers all cards to be a string of characters and takes no recognition of card contents. For convenience, you might assign a different UPDATE deck name to each routine, but there is no requirement to do so. UPDATE divides text cards into decks, following directive instructions.

You control the order of decks in the program library; decks appear in the order in which they are found in the input stream. A common deck must precede any regular deck that might call the common deck.

All cards following a DECK or COMDECK directive, until the next DECK or COMDECK directive, are considered to be part of the deck, and each receives a unique sequence number. The directive defining the deck itself is assigned a sequence number 1. Any READ directive among the text cards causes UPDATE to temporarily stop reading from the current input stream and to read from the specified file until an end-of-file is encountered; reading then resumes from the main input stream. Text cards read from the file specified by READ are numbered as if they were part of the original input text.

CARD IDENTIFICATION

The image of each card stored in a deck contains information known as correction history bytes. This information, generated by UPDATE, maintains the history and status of a card and is the means by which UPDATE can reverse status. Deletion of a card, for example, is accomplished by the addition of a correction history byte to the card image, rather than by a physical deletion of the image. Consequently, the card can be reactivated at some later time. Only purge operations are irreversible.

A DECK or COMDECK directive is written to the program library as part of the deck text. Consequently, these directives can be referenced just as can any other card in the text. Deactivating a DECK directive, for example, has the effect of making its following text a part of the deck that precedes it in the library.

UPDATE recognizes one full form and two short forms of card identifiers. The full form card identifier is as follows:

ident.seqnum

ident. One- through eight-character name of a correction set or deck. A period terminates the ident name.

seqnum Decimal ordinal (1 through 65535) representing the sequence number of the card within the correction set or deck. Any character other than 0 through 9 terminates the sequence number.

The two short forms of card identifiers can be used on INSERT or DELETE directives. The short forms are expanded as follows:

seqnum Expands to idname.seqnum; idname is a correction set identifier, whether or not it is also a deck name.

.seqnum Expands to dname.seqnum; dname is a deck name.

In the short form, idname is assumed to be the last explicitly named ident given on an INSERT or DELETE directive, whether or not it is a deck name. The dname is assumed to be the last explicitly named ident given on an INSERT or DELETE directive that is known to be a deck name. Both of these default idents are originally set to YANK\$\$\$, so the first directive using a card identifier must use the full form to reset the default.

All deck names are also idents (but all idents are not decks). Thus, if EXAMPLE is the deck name last used and there is no subsequent explicit reference to a correction set identifier, both .281 and 281 expand to EXAMPLE.281 as the identifier. If there is an explicit reference to a correction set identifier after the explicit reference to the deck name, 281 expands to the correction set identifier, while .281 expands to EXAMPLE.281 as the identifier.

Figure 5-3 shows differences in identifier expansion, depending on the order of directive records, assuming that A is a deck name and B is a correction set identifier on an UPDATE old program library.

```

*ID C
*INSERT A.2
  data card
*INSERT B.1
  data card
*D 2, 3      expands to *DELETE B.2, B.3
*D 4, .5    expands to *DELETE B.4, A.5
*D .7, 5    expands to *DELETE A.7, B.5
*D .9, .10  expands to *DELETE A.9, A.10

whereas:

*ID D
*INSERT B.1
  data card
*INSERT A.2
  data card
*D 2, 3      expands to *DELETE A.2, A.3
*D 4, .5    expands to *DELETE A.4, A.5
*D .7, 5    expands to *DELETE A.7, A.5
*D .9, .10  expands to *DELETE A.9, A.10

```

Figure 5-3. Card Identifier Expansion

CORRECTION RUN

A correction run, which is the most common use of UPDATE, introduces changes into the existing program library. UPDATE recognizes a correction run, as opposed to a creation run, under either of the following circumstances:

- The first directive, other than a comment, is IDENT.
- The first directive, other than a comment, is READ or ADDFILE and the first directive on the alternative file is IDENT (in the case of READ) or DECK or COMDECK (in the case of ADDFILE).

All directives can be used during a correction run.

The IDENT directive establishes a name for the correction set. Any cards inserted into the library are sequenced within this name. On subsequent correction runs, individual cards in the correction set can be referenced by sequence number. The entire correction set can also be referenced as a whole.

When a new program library is being generated, all corrections must be part of a correction set, with the exception of ADDFILE, MOVE, PURDECK, and PURGE. That is, IDENT must be the first directive other than a comment. If READ is the first directive, the alternative input file must have IDENT as its first directive. If a new program library is not being generated (that is, routines are being extracted, but no changes made), directives can appear without a correction set identifier.

The following directives need not be part of a correction set. They are directives PURGE, PURDECK, and ADDFILE (which cause the current set to be terminated) and COMPILE and MOVE. The COMPILE and MOVE directives, like the comment directive, can appear anywhere inside or outside a correction set. A COMPILE directive is not processed until all corrections have been made. A MOVE directive is processed immediately.

More than one correction set can be introduced during a single run. The correction set established by the first IDENT directive remains in effect until UPDATE either encounters another IDENT directive or encounters a PURGE, PURDECK, or ADDFILE directive. The subsequent IDENT directive establishes a second correction set name.

A correction run can include the addition of new decks to the program library when a new program library is created. Decks to be added are identified by a DECK or COMDECK directive following an INSERT, DELETE, or ADDFILE directive.

Deck List and Directory Order

UPDATE maintains a list of all decks in the program library, known as a deck list. The order of entries in the deck list is under your control; original deck list entries correspond to the order in which decks are written during the creation run. Subsequent additions of decks are made at the location you specify by using a preceding INSERT, DELETE, or ADDFILE directive.

The location of an entry in the deck list is significant in terms of parameters for PURDECK, YANKDECK, and COMPILE directives, in which a range of decks is referenced. The order of names in a range reference must be the same as the order in the deck list. The decks named and all those between are then processed in accordance with the directive. An error exists if they are in reverse order.

Similarly, as each correction set is introduced into the program library, UPDATE creates an entry in an internal directory in chronological sequence. The location of an entry in the directory is significant in terms of parameters for PURGE and YANK, directives in which a range of correction sets is referenced. The order of reference must be the same as the order of the directory. The identified correction sets and all those between are processed in accordance with the directive. An error exists when a correction set range is not referenced in the order in which the sets were introduced into the library.

Purge and Yank Directives

The two purge directives are PURGE and PURDECK. PURGE operates on all cards identified by correction set name, while PURDECK operates on all cards within an identified deck. (Introduction of a new deck on a creation run must be made as part of a correction set, but that addition usually is the only change within that correction set.) The two yank directives are YANK and YANKDECK. As with the purge directives, the former operates with correction sets and the latter with decks.

The purge directives differ from the yank directives in that yank operations are temporary. Cards yanked from the program library are temporarily deactivated. They can be reactivated by a subsequent yank, delete, or purge of the yank directive that inactivated the cards.

In contrast, any change made to a program library through a purge directive is permanent. A reversal of a purge operation is possible only through the reintroduction of the cards into the library, as if they had not previously existed.

Since the YANK directive itself must be introduced as part of a correction set, a future correction set that deactivates the cards in the correction set containing the YANK reactivates the cards in the original correction set. The following are examples.

To inactivate all cards added by IDENT PSR003:

```
*IDENT TAKEOUT
*YANK PSR003
```

To reactivate the same cards:

```
*IDENT PUTBACK
*YANK TAKEOUT
```

Overlapping correction messages might be produced as a result of these procedures.

UPDATE stores all YANK directives in a deck having the deck name YANK\$\$\$\$. The YANK\$\$\$ deck cannot be referenced on UPDATE directives. Individual cards in the YANK\$\$\$ deck can be deactivated by another yank, as in the previous example, or by a delete or purge. For example, the card *YANK PSR003 in the previous example can be deleted by the following:

```
*IDENT PUTBACK
*D,TAKEOUT.1
```

Cards in a correction set that have been yanked are physically present but logically inactive. A reference to a card in a yanked correct set generates an advisory message indicating an overlapping correction.

Overlapping Corrections

UPDATE can detect four overlapping correction situations. When any of these types is detected, UPDATE prints the offending line with the words TP.n OVLP appended on the far right.

<u>Type</u>	<u>Meaning</u>
1	Two or more modifications are made to one card during a single update run.
2	A modification attempts to activate an already active card.
3	A modification attempts to deactivate an already inactive card.
4	A card is inserted after a card that was inactive on the OLDPL.

Detection of an overlap does not necessarily indicate a user error. Overlap messages are advisory, and they point to conditions in which the probability of error is greater than normal.

Type TP.2 and TP.3 are detected by comparing existing correction history bytes with those to be added. Complex operations involving YANK and PURGE might generate these overlap messages even though no overlap occurs.

Modifications for each correction set are performed by UPDATE in the order in which sets are introduced. The order is irrelevant if no correction is dependent on another. If a dependent relationship exists, however, order is of paramount importance.

UPDATE DIRECTIVES

Directives are instructions for UPDATE to follow in creating its output files. A directive must begin in column 1 of the card with the master control character. Each directive has both a full keyword and an abbreviated keyword, as shown in table 5-2.

The general format is as follows:

*keyword p-list

* Master control character that distinguishes a directive from a text card. Must appear in column 1. This character can be changed through the *=c parameter of the UPDATE control statement.

keyword Name of one of the UPDATE directives or an abbreviation for a directive. No blanks can occur between the master control character and the keyword; a comma or blank terminates the keyword.

p-list Parameters identifying decks, cards, or files. Multiple blanks can appear between the keyword and parameters. Parameters in the list are separated by commas; embedded blanks cannot appear in the list.

Notice that several parameters contain a period as part of a single parameter.

No terminator appears at the end of a directive.

The master control character is recorded in the program library. For a correction run, the master control character should match the character used when the program library was created; if the characters do not match, UPDATE uses the character stored as part of the program library.

Any card in the input stream that cannot be recognized as a directive or as a comment is assumed to be text.

The directives are described in alphabetical order.

ADDFILE DIRECTIVE

The ADDFILE directive causes UPDATE to add a file of new decks to the new program library. It differs from the READ directive in that contents of the specified file are limited to those that add decks. The first card of the specified file must be a DECK or COMDECK directive. No directives other than comments, DECK, or COMDECK can appear in the file.

The ADDFILE directive format is shown in figure 5-4. If only one parameter appears, it is assumed to be lfn.

<u>Add after card identified</u>	
*ADDFILE lfn,ident.seqnum	
lfn	Name of file from which decks to be added are taken. Default is the file specified by the I parameter of the UPDATE call.
ident.seqnum	Identifier of card after which decks are to be placed on the program library. If omitted, the addition is made at the end of the program library.
<u>Add after deck identified</u>	
*ADDFILE lfn,deck	
lfn	Name of file from which decks to be added are taken. Default is the file specified by the I parameter of the UPDATE call.
deck	Name of deck after which decks are to be placed on the program library. If omitted, the addition is made at the end of the program library.

Figure 5-4. ADDFILE Directive Format

When the specified file is not INPUT, UPDATE reads directives and text cards until an end-of-file (#1C) is encountered. UPDATE then returns to the file specified by the I parameter of the UPDATE call and continues processing the main input stream. When the file specified on the ADDFILE directive is INPUT, however, UPDATE reads directive and text cards only until either an end-of-file or an UPDATE directive other than DECK, COMDECK, or CALL is encountered.

An ADDFILE directive cannot appear among directives read from a file specified by a READ directive; otherwise, it can appear anywhere in the input stream, but its appearance terminates the current correction set.

BEFORE DIRECTIVE

The BEFORE directive adds the text cards that follow it to the program library before the card specified.

New cards receive card identifiers established by the correction set name of the preceding IDENT directive.

The BEFORE directive format is shown in figure 5-5.

*BEFORE ident.seqnum	
ident.seqnum	Card identifier of the card before which the insertion is to be made.

Figure 5-5. BEFORE Directive Format

CALL DIRECTIVE

The CALL directive causes UPDATE to write the text of a previously encountered common deck onto the compile file. The directive itself is stored as part of a deck and can be referenced by its sequence number. It is effective only within a deck.

The CALL directive format is shown in figure 5-6.

*CALL deck	
deck	Name of an existing common deck to be written to the compile file.

Figure 5-6. CALL Directive Format

Neither the CALL directive nor the COMDECK directive, which defined the deck, becomes part of the compile file.

A common deck can call other common decks, but it must not call itself or call a deck that contains a call to the common deck.

COMDECK DIRECTIVE

The COMDECK directive establishes a common deck that can be called from other decks as they are being written to the compile file. It is one of the two directives that establish the existence of a creation run. The directive can be used in any correction run to add a common deck to a particular location in the program library.

The COMDECK directive format is shown in figure 5-7.

*COMDECK deck,NOPROP	
deck	Name of common deck being added. Must consist of one through eight characters, A through Z, 0 through 9, or + - / * () \$ = _ . Must not duplicate the name of an existing deck.
NOPROP	Indicates that decks calling this common deck are not to be considered as modified when the common deck itself is modified; that is, the effects of common deck changes are not to be propagated during a normal UPDATE mode. Optional.

Figure 5-7. COMDECK Directive Format

The COMDECK directive itself is part of the program library and has a sequence number of 1 within the name established by the directive. For a creation run, the deck order in the input stream determines the location of the common deck in the program library. For a correction run, the location in the program library is determined by the preceding INSERT directive or by the location resulting from a preceding DELETE or ADDFILE. Common decks need not appear first on the program library, but they must appear before any decks from which they are called during a creation run.

The NOPROP parameter of the COMDECK directive that created a common deck determines whether a deck calling a corrected common deck will also be considered corrected.

COMPILE DIRECTIVE

The COMPILE directive affects the decks to be written to the compile file and to any new program library or source file during normal or quick UPDATE mode. The directive is ignored during a full UPDATE. Compile directive processing in normal and quick modes is as follows:

<u>Mode</u>	<u>Description</u>
Normal	Decks specified on COMPILE directives and corrected decks are written to the compile file.
Quick	Decks specified on COMPILE directives and any common decks they call are written to the compile file.

The COMPILE directive format depends on whether decks to be written are specified individually by name or as a range of deck names, as shown in figure 5-8.

<u>Compile listed decks</u>	
*COMPILE deck1,deck2, . . . ,deckn	
deck	Name of deck to be written to the compile file, new program library file, and source file.
<u>Compile range of decks</u>	
*COMPILE deck1.deck2	
deck1.deck2	Names of first and last decks in range, inclusive, to be written to the compile file. The name of deck1 must appear in the old program library deck list before deck2.

Figure 5-8. COMPILE Directive Format

Decks are written to the compile file in the order in which they exist on the old program library, except in K mode. When K is specified on the UPDATE control statement, decks are written in the order of their occurrence on COMPILE directives. If COMPILE directives specify two different orderings for a deck or decks, the latest ordering is used. K mode is ignored during a full update. Decks are always written to the new program library and the source file in the order in which they exist on the old program library, regardless of any compile file ordering during a K mode update. If K is specified for a normal mode update (neither F nor Q specified) any corrected decks not mentioned on COMPILE directives are written at the end of the compile file.

COMPILE directives can appear anywhere within the input stream. They do not affect the current correction set name.

The COMPILE directive also affects the contents of any new program library and source file, as shown in table 5-3.

Table 5-3. File Contents and Update Mode

File	Normal Mode Contents	Full Mode Contents	Quick Mode (Q) Contents
New program library	All regular and common decks after corrections made in library.†	Same as normal mode source file.	Decks specified on COMPILE directives, any common decks they call, and any common decks encountered on old program library prior to all decks of COMPILE or ADDFILE.
Compile File	All decks corrected or listed on COMPILE directives, and any deck calling a corrected common deck (unless NOPROP specified on COMDECK).	All active decks on old program library.	All decks on COMPILE or ADDFILE directives and any common decks they call.
Source File	All currently active DECK, COMDECK, and CALL directives and active text required to recreate library.	Same as normal mode source file.	Currently active cards required to create new program library resulting from quick mode.

†T parameter excludes common decks.

DECK DIRECTIVE

The DECK directive establishes a deck in the program library. It is one of the two directives that establish the existence of a creation run. The directive also can be used in any correction run to add a deck to the location indicated by a preceding ADDFILE directive.

The DECK directive format is shown in figure 5-9.

Each deck must have a unique name within the program library.

The DECK directive itself is part of the program library and has a sequence number of 1 within the name established by the directive.

*DECK deck	
deck	Name of deck. Must consist of one through eight characters, A through Z, 0 through 9, or + - / * () \$ = _ . Must not duplicate the name of any other deck in program library.

Figure 5-9. DECK Directive Format

DEFINE DIRECTIVE

The DEFINE directive establishes a name to be tested by IF,DEF or IF,-DEF. The names on a DEFINE DIRECTIVE are unrelated to ident or deck names. DEFINE directives must appear in a correction set and can be placed anywhere in a correction set. DEFINE directives are placed in the YANK\$\$\$ deck on the program library.

The format of the DEFINE directive is shown in figure 5-9.1.

<pre>*DEFINE,name1,name2,...namen</pre> <p>name Name for testing by IF,DEF or IF,-DEF directives.</p>
--

Figure 5-9.1 DEFINE Directive Format

DELETE DIRECTIVE

The DELETE directive deactivates a card or group of cards and optionally adds text cards following the directive. A deactivated card remains on the library unless it is removed by editing. It retains its sequencing and can be referenced just as if it were not deactivated. A deactivated card is not written to any compile file or source file, however.

The DELETE directive format depends on whether cards to be deactivated are specified by card identifier or by a range of cards, as shown in figure 5-10.

<p>Delete specified card</p> <pre>*DELETE ident.seqnum</pre> <p>ident.seqnum Card identifier for single card to be deleted.</p> <p>Delete range of cards</p> <pre>*DELETE ident1.seqnum,ident2.seqnum</pre> <p>ident1.seqnum, Card identifiers of first and last cards, inclusive, in ident2.seqnum sequence of cards to be deleted. Card ident1.seqnum must appear before ident2.seqnum in the existing library. The range can include cards in a deactivated state.</p>
--

Figure 5-10. DELETE Directive Format

ENDIF DIRECTIVE

The ENDIF directive is used to indicate the end of conditional text. ENDIF is used when the numbers of lines to skip is omitted from the IF directive. If num is given on the IF directive, an ENDIF appearing within the range of num can cause the wrong number of lines to be skipped, or the ENDIF directive to be written to the compile file.

The format of the ENDIF directive is shown in figure 5-10.1.

```
*ENDIF
```

Figure 5-10.1 ENDIF Directive Format

IDENT DIRECTIVE

The IDENT directive establishes the name for the set of corrections being made. Cards added in this correction set are sequenced within the name specified. Any card whose status is changed by this set receives a correction history byte that references the name from IDENT. All correction set names must be unique.

The IDENT directive format is shown in figure 5-11.

```
*IDENT ident,B=num,K=ident,U=ident

ident      Name to be assigned to this correction set. Must consist of one
           through eight characters, A through Z, 0 through 9, or + - / * ( )
           $ = _ . Must not duplicate the name of another correction set or
           deck.

B=num      Bias to be added to sequence numbers within deck.

K=ident     Indicates that specified correction set name must exist in the
           directory of the library before corrections can be made.

U=ident     Indicates that specified correction set name must not exist in the
           directory of the library.
```

Figure 5-11. IDENT Directive Format

The B, K, and U parameters can appear in any order. More than one K or U parameter can be specified; in this instance, all correction set names specified must meet the criteria before the correction set is processed. If the criteria of these parameters are not met, UPDATE skips the correction set and resumes processing with the next IDENT, PURGE, PURDECK, or ADDFILE directive.

IF DIRECTIVE

The IF directive conditionally writes text to the compile file. When UPDATE encounters an IF directive, the text following the directive is written or skipped depending on the condition.

The format of the IF directive is shown in figure 5-11.1.

<u>Name must be known in run</u>	
*IF type,name,num	
type	Type of condition.
DECK	Name is a deck name.
IDENT	Name is an ident name.
DEF	Name is defined by a DEFINE directive.
name	Deck name, ident name or defined name according to type.
num	Number of active line images to be skipped (not written to compile file) if name is not known. Optional. If num is omitted, and the name is not known, lines will be skipped until an ENDIF directive is encountered.
<u>Name must be unknown in run</u>	
*IF -type,name,num	
type	Type of condition.
DECK	Name is a deck name.
IDENT	Name is an ident name.
DEF	Name is defined by a DEFINE directive.
name	Deck name, ident name or defined name according to type.
num	Number of active line images to be skipped (not written to compile file) if name is known. Optional. If num is omitted, and the name is known, lines will be skipped until an ENDIF directive is encountered.

Figure 5-11.1 IF Directive Format

INSERT DIRECTIVE

The INSERT directive adds text cards following it to the program library at the location specified.

The INSERT directive format is shown in figure 5-12.

New cards receive card identifiers established by the correction set name of the preceding IDENT directive.

```
* INSERT ident.seqnum
      ident.seqnum Card identifier of card after which the insertion is to be made.
```

Figure 5-12. INSERT Directive Format

MOVE DIRECTIVE

The MOVE changes the order of the existing decks on the program library.

A MOVE directive can appear anywhere in the input stream. It does not terminate insertions or the current correction set.

The YANK\$\$\$ deck containing all YANK directives in a deck cannot be moved.

A MOVE directive cannot reference a deck added in the same run.

A MOVE directive takes effect before any modifications entered in the same run.

The MOVE directive format is shown in figure 5-13.

```
*MOVE deck1,deck2
deck1      Name of deck to be moved.
deck2      Name of deck after which the moved deck is to be placed.
```

Figure 5-13. MOVE Directive Format

PURGE DIRECTIVE

The PURGE directive permanently removes a correction set or group of correction sets from the program library. Every card in the correction set is purged, regardless of its status as active or inactive. Purging, unlike yanking, cannot be rescinded.

The PURGE directive format depends on whether correction sets to be purged are specified individually by correction set name, by a range of correction set names, or by relative time of introduction into the program library, as shown in figure 5-15.

A PURGE directive can appear anywhere in the input stream, but it terminates the current correction set. Any directive following PURGE must begin a new correction set.

<u>Purge listed correction sets</u>	
<code>*PURGE ident1,ident2, . . . ,identn</code>	
ident	Identifies a correction set to be purged. Identifiers can appear in any order.
<u>Purge range of correction sets</u>	
<code>*PURGE ident1.ident2</code>	
ident1.ident2	Identify first and last correction sets, inclusive, to be purged. Identifiers must appear in the relative order in which the correction sets were introduced into the program library; that is, they must appear in the order in which they exist in the directory.
<u>Purge later correction sets</u>	
<code>*PURGE ident,*</code>	
ident	Identifies correction set to be purged along with all correction sets introduced after the one specified.
*	Indicates that the program library is to return to an earlier level. Intervening purge directives prevent complete return.

Figure 5-15. PURGE Directive Format

READ DIRECTIVE

The READ directive causes UPDATE to temporarily stop reading the current input stream and to begin reading an input stream from the file specified on the READ directive. READ differs from ADDFILE in that the content of the file specified by READ is not restricted except to prohibit the appearance of either another READ directive or an ADDFILE directive. UPDATE reads from the specified file until an end-of-file (#IC) is encountered. Processing then continues with the main input stream.

The READ directive format is shown in figure 5-16.

<pre>*READ lfn</pre>	
lfn	Name of alternate file containing input stream.

Figure 5-16. READ Directive Format

WIDTH DIRECTIVE

The WIDTH directive overrides the default compile file line image width settings as specified by default or by D and/or 8 on the UPDATE control statement. The format for the WIDTH directive is shown in figure 5-16.1.

<pre>*WIDTH datlen,idlen</pre>	
datlen	Number of characters of line image text that is written.
idlen	Width of the identification field following the line image. If the idlen is too small to hold the full identifier name and sequence number, the sequence number overwrites the identifier name.

Figure 5-16.1. WIDTH Directive Format

The sum of the length of datlen and idlen must be less than or equal to 256 characters. If idlen is set to zero, the identification field is suppressed. If *WIDTH is specified with no parameters, the run default settings are restored. If only the length of the identification field is specified (*WIDTH ,idlen), then datlen is the previous setting used. If only datlen is specified (*WIDTH datlen), the previous setting of idlen is used.

YANK DIRECTIVE

The YANK directive temporarily removes a correction set or group of correction sets from the program library. Cards activated by the correction set are deactivated; cards deactivated by the correction set are reactivated. YANK differs from PURGE in several respects: YANK must be part of a correction set; it does not terminate the current correction set; its effects can be rescinded.

If the library has been edited, it may not be possible to rescind the effects of a YANK.

The YANK directive format depends on whether correction sets to be yanked are specified individually by correction set name or by a range of correction set names, as shown in figure 5-17.

UPDATE places the YANK directive in the YANK\$\$\$ deck. If a correction has been yanked, it is ignored during compile file or source file generation.

<u>Yank listed correction sets</u>	
*YANK ident1,ident2, . . . ,identn	
ident	Identifies a correction set to be yanked. Identifiers can appear in any order.
<u>Yank range of correction sets</u>	
*YANK ident1.ident2	
ident1.ident2	Identifies first and last correction sets, inclusive, to be yanked. Identifiers must appear in the relative order in which the correction sets were introduced into the program library; that is, they must appear in the order they exist in the directory.

Figure 5-17. YANK Directive Format

YANKDECK DIRECTIVE

The YANKDECK directive temporarily deactivates all cards within the decks specified. All cards are deactivated, regardless of the correction set to which they belong. YANKDECK differs from PURDECK in several respects: YANKDECK must be part of a correction set; it does not terminate the current correction set; its effects can be rescinded.

If the library has been edited, it may not be possible to rescind the effects of a YANK.

The YANKDECK directive format is shown in figure 5-18.

*YANKDECK deck1,deck2, . . . ,deckn	
deck	Name of deck to be yanked. Names can appear in any order.

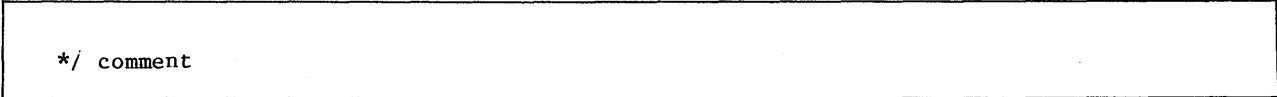
Figure 5-18. YANKDECK Directive Format

The deck YANK\$\$\$ cannot be deactivated as a whole. Individual YANK directives within this deck can be yanked by a YANK directive, however.

/ COMMENT DIRECTIVE

The / comment directive introduces a comment into the listable output file. UPDATE ignores this card except to copy it to the output file. A comment can appear at any place in the input stream.

The / comment directive format is shown in figure 5-19. The slash must appear in column 2. Column 3 must be a comma or a blank. The slash can be redefined as another character through the /=c parameter of the UPDATE call.



```
*/ comment
```

Figure 5-19. / Comment Directive Format

UPDATE CONTROL STATEMENT

The format of the control statement that calls UPDATE to execution is shown in figure 5-20. All parameters are optional and can appear in any order. A comma must separate parameters.

Update, { C=file } ,D,E,F, { I=1fn } , { K=1fn } ,L=opt, { N=1fn/#nnn } , { O=1fn/#nnn } ,
 { P=1fn } ,Q, { S=1fn/#nnn } , { T=1fn/#nnn } ,U,8,*=c,/=c.

C	Compile file name. The content of the compile file is determined by the UPDATE mode.
	omitted Decks are written to file named COMPILE.
	or C
	C=1fn Decks are written to file named 1fn. File length is
	or #100 small pages or the number of pages specified by
	C=1fn/#nnn nn.
	C=PUNCH Decks are written to file named PUNCH. The D and 8
	parameters are implied.
	C=0 Compile file suppressed.
D	Data width on compile file excluding UPDATE sequence identifiers.
	omitted 72 columns of data.
	D 80 columns of data.
E	Edit mode.
	omitted No editing is performed.
	E New program library contains only active cards. Idents
	from which inactive cards have been removed are marked
	with an E.
F	Full UPDATE mode.
	omitted Normal selective UPDATE mode, as long as Q is not speci-
	fied. The compile file contains only those decks corrected
	in this run or otherwise specified on COMPILE directives.
	F Full UPDATE mode. The compile file contains all active
	decks in the program library. F overrides K.
G	Pullmod file name.
	omitted No pullmod file will be written.
	G Recreated idents will be written to the file specified
	by the S parameter, if S is present. If S not present,
	idents will be written to the file named SOURCE.
	G=1fn Recreated idents will be written to file named 1fn.

Figure 5-20. UPDATE Control Statement Format (Sheet 1 of 5)

I	Input stream file name.	
	omitted or I	Directives and text are on the file named INPUT.
	I=lfm	Directives and text are on file named lfm.
K	Write compile file in the order in which compile directives are encountered in the input file.	
	omitted	Decks on the compile file are ordered as they exist on the old program library. The compile file is COMPILE or the file specified by C=lfm.
	K	Decks on the compile file (COMPILE or the file specified by C=lfm) are ordered by their occurrence on COMPILE directives in the input stream.
	K=lfm	Decks are written to the file named lfm in COMPILE directive order. K=lfm overrides C=lfm.
L	Listable output options to be written to file named with the O parameter.	
	omitted	For a creation run, options A, 1, and 2. For a correction run, options A, 1, 2, 3, and 4.
	L=c...c	Each character in string c...c selects one of the following options:
	A	Error decks, correction set identifiers, common decks, and decks written to the compile file are listed.
	F	Full listing.
	0	The character 0 overrides any other options specified and suppresses the entire listing.
	1	List all input lines in error and associated error messages.
	2	List all directives from the input file with ***** preceding each valid directive. List active CALL directives encountered on the old program library. Directives from the old program library are preceded by the deck name in which the directive appears.

Figure 5-20. UPDATE Control Statement Format (Sheet 2 of 5)

3 Comment on each card changed. Comments include the deck name, card image, card identifier and sequence number, and an indicator of action taken for that card:

I Card added.

A Inactive card reactivated.

D Active card deactivated.

P Card purged. If the card was active, ACTIVE also appears.

4 List text cards of input stream established by directives. Cards read as a result of a READ directive are identified to the right with the file name; cards inserted as a result of an ADDFILE directive are listed only when option 4 is explicitly selected. *ERROR* accompanies any cards in error.

5 List all active DECK, COMDECK, and CALL directives encountered on the old program library that are preceded by the name of the deck in which the directive appears.

7 List all active cards (options 2, 3, 4, and 5 override option 7).

9 List all active and inactive cards with status.

I Inactive

A Active

Option 3 overrides option 9.

L=0 Suppress all listings.

N New program library file name.

omitted In a correction run, suppress new program library generation. In a creation run, write a new program library on the file named NEWPL.

N Write new program library on file named NEWPL.

N=lfm Write new program library on file named lfm. File length is #100 512-word blocks or the number of blocks specified by nnn.
or
N=lfm/#nnn

Figure 5-20. UPDATE Control Statement Format (Sheet 3 of 5)

O	Listable output file name.	
	omitted or O	Write list output to file named OUTPUT.
	O=lfn or O=lfn/#nnn	Write list output to file named lfn. File length is #100 512-word blocks or the number of blocks specified by nnn.
P	Old program library file name.	
		The P parameter is valid only for a correction run.
	omitted or P	Old program library resides on file named OLDPL.
	P=lfn	Old program library resides on file named lfn.
Q	Quick UPDATE mode. The source file and the new program library are described in table 5-3.	
	omitted	Normal selective UPDATE mode when F is also omitted.
	Q	Only those decks specified on COMPILE directives are processed. Corrections to decks not specified on COMPILE directives are not processed; the corrections normally produce a fatal error (refer to the U parameter description). The unprocessed corrections are listed, and no compile, source, or new program library is produced. The compile file contains only decks referenced on COMPILE directives and the common decks they call.
		The Q parameter takes precedence when both F and Q are specified.
S	Source file name. The content of this file is determined by the UPDATE mode.	
	omitted	Suppress source output file unless it is selected by the T parameter.
	S	Source output file to be written on file named SOURCE.
	S=lfn or S=lfn/#nnn	Source output file to be written on file named lfn. File length is #100 512-word blocks or the number of blocks specified by nnn.

Figure 5-20. UPDATE Control Statement Format (Sheet 4 of 5)

T Omit common decks from source file. The content of the source file is determined by the UPDATE mode, with the T parameter excluding common decks.

omitted Suppress source output unless it is selected by the S parameter.

T Source output file to be written on file named SOURCE, with common decks excluded.

T=lfm Source output file to be written on file named lfm, with
or common decks excluded. File length is #100 512-word
T=lfm/#nnn blocks or the number of blocks specified by nnn.

The T parameter takes precedence over the S parameter.

U Override abort for unprocessed modifications.

omitted In Q mode, corrections to decks not specified on COMPILE directives cause the update run to abort. No compile, source, or new program library is produced.

U Corrections to decks not specified on compile directives are listed as errors, but the run does not abort and all output files are produced. The termination value for the run is 4.

8 Card image width on compile file, including UPDATE sequence identifiers.

omitted 90-column card image, which preserves columns 73 through 80 of original card.

8 80-column card image, with UPDATE sequence information in columns 73 through 80.

* Master control character for directives.

omitted * is the first character of each directive.

*=c c is the first character of each directive for this UPDATE run. c can be any character, A through Z, 0 through 9, or + - * / \$ or =. If the character specified for a correction run is not the same as the character used when the old program library was created, the old program library character is used.

/ Comment control character in column 2.

omitted Comment control character is /.

/=c c is the comment control character. c can be any character, A through Z, 0 through 9, or + - * / \$ or =.

Figure 5-20. UPDATE Control Statement Format (Sheet 5 of 5)

DEBUG, LOOK, and DUMP are utilities for testing and debugging a correctly compiled or assembled program that executes unsatisfactorily. These utilities can be executed either interactively or in batch mode.

Differences among these three utilities include the following:

- DEBUG displays or alters the contents of selected locations during program execution. It is valid only with controllee files.
- LOOK displays or alters the contents of selected locations in any type of file. It can be used with controllee files, drop files, or data files. Its most common use is through an interactive terminal.
- DUMP displays a preselected set of elements from a drop file.

Both LOOK and DEBUG use a set of directives supplied by a programmer to receive detailed control information. A batch job must have the directives on a file available to the job.

You can enter directives interactively and receive output as it is generated in response to the directive. Output from most directives is returned to the terminal; some directives can specify a file to receive output. When the utility is ready for another directive, the character ? appears at the terminal. Directives must be entered on a single line.

Typical use of the debugging utilities involves using LOOK to edit a FORTRAN source program interactively until the program compiles successfully; executing the compiled program and possibly receiving a dump on a fatal error condition, or else possibly forcing such a dump by making a DUMP request; using DEBUG to observe intermediate program values during reexecution of the program under DEBUG control; and subsequently using DEBUG or LOOK to modify the program until it executes satisfactorily.

DEBUG

Through DEBUG you can set instruction breakpoints and a memory reference breakpoint in a program and then issue EXECUTE and CONTINUE directives to step through the execution of the program from one breakpoint to the next. At each breakpoint, current values of variables in the program can, for example, be dumped. DEBUG can also be used to modify, display, and dump user registers and areas in virtual memory designated by hexadecimal addresses.

An FTN200 program being executed under DEBUG must have been compiled without the SDEB compile option or with SDEB=0 if symbolic addresses--labels, names, line numbers--are to be used in the DEBUG directives. DEBUG executes entirely within your virtual space, starting at hexadecimal virtual bit address #7FFF00000000 and extending upward; therefore, the program being debugged must not use or reference this area.

Modules that reside in a user dynamic library or the system shared library, SHRLIB, can be debugged and altered with no effect on the other users of the shared library. The shared library resides in virtual memory above #800 000 000 000 and below #C00 000 000 000. If a breakpoint is set in this region, it has no effect on other users of the system shared library. When a controllee that uses the SHRLIB is debugged, DEBUG issues the following message:

WARNING. CONTROLLEE REQUIRES DYNAMIC LINKING

After the DEBUG control statement is issued, DEBUG remains in execution until an EXECUTE, STEP, or CONTINUE directive causes it to relinquish control to the user program. Control does not return to DEBUG until a user-specified breakpoint is reached during execution. When the user program terminates, DEBUG terminates also; more DEBUG directives can be processed only after another DEBUG control statement has been issued.

NOTE

When using DEBUG, SIL modules used dynamically may be linked in a different order than that found on the dynamic load map produced by DUMP. Your controllee may fail in a different order due to a possible reordering of dynamic modules if DEBUG uses them before the program does.

The dynamic load map that is displayed by DUMP when a dynamically loaded controllee aborted in execution should not be used if the same controllee is executed again under the control of DEBUG, as the address of the dynamically linked modules might change.

When running DEBUG in interactive mode, the program being debugged is supplied with the parameters normally provided on the execute line. However, in batch mode there is no mechanism for providing those parameters. Debugging in batch mode, therefore, is possible only for programs that can execute meaningfully by using built-in default values for all parameters.

At security-sensitive sites with production user numbers and production files (refer to chapter 7 of the Installation Handbook for details), DEBUG cannot be used by production user numbers on production controllees.

DEBUG CONTROL STATEMENT

The control statement that initiates execution of DEBUG is shown in figure 6-1. The parameter fname must always be the first parameter, but the order of the I= and O= parameters can be reversed.

NOTE

Regarding the selection of the iname and oname parameters: When invoking DEBUG, the files specified by the iname or oname parameters (or the respective default file names) are opened by the DEBUG processor in its initialization phase. The controllee that is subsequently invoked under control of DEBUG must not attempt to open and use files that conflict with the files specified by those parameters. This often happens if an FTN200 controllee, which writes to file OUTPUT, for example, is invoked under the control of DEBUG without an oname specified on the command line (oname, therefore, defaults to OUTPUT). This will result in the FTN200 program failing when it tries to open file OUTPUT again in its initialization, after DEBUG has already opened OUTPUT. Note that DEBUG always opens oname, even if it never has to write anything to that file.

DEBUG, fname, I=iname, O=oname/olen.

fname Name of the existing permanent or local file that is to be the controllee file for DEBUG. It must be a virtual code file produced by the LOAD utility, and you must have both read and execute access permissions for the file.

I=iname For batch mode only, a file containing the DEBUG directives. If I=iname is omitted, directives are read from INPUT.

O=oname/olen For batch mode, the file to which all DEBUG output is written; for interactive mode, the file to which data generated by the SNAP command is written. olen is the length of the output file in 512-word blocks.

The default file is OUTPUT. The default file length is #100 512-word blocks.

If the user program opens the file OUTPUT, oname must not be output.

Figure 6-1. DEBUG Control Statement Format

The following are sample DEBUG control statements.

- DEBUG(MYCTEE)
- DEBUG(MYCTEE,I=MYINP,O=MYOUT/#2C3)
- DEBUG(MYCTEE,O=MYOUT,I=MYINP)

DEBUG DIRECTIVES

The general format of each DEBUG directive is as follows:

directive,parameter-set

Each directive name can be abbreviated to the minimum character string needed to distinguish it from all other directive names. The minimum abbreviation for each directive is underlined in figure 6-2.

Parameters are positional and can be separated from each other and the directive name by either a blank or a comma. A null parameter must be indicated by commas delimiting its position.

DEBUG directives are listed in alphabetical order in figure 6-2. The subsequent directive descriptions are grouped according to a common function.

<u>Directive</u>	<u>Description</u>
<u>ASCII</u>	Enter data in ASCII form.
<u>BACK</u>	Display the data preceding the last display location.
<u>BKPT</u> or <u>BKPTR</u>	Set or remove breakpoints.
<u>CONTINUE</u>	Continue execution from the last user breakpoint.
<u>DDECIMAL</u>	Display data in hexadecimal and decimal.
<u>DDREG</u>	Display register contents in hexadecimal and decimal.
<u>DECIMAL</u>	Enter data in decimal form.
<u>DFLOAT</u>	Display data in hexadecimal and floating point.
<u>DFREG</u>	Display register contents in hexadecimal and floating point.
<u>DISPLAY</u>	Display data in hexadecimal and ASCII.
<u>DREG</u>	Display register contents in hexadecimal.
<u>EDREG</u>	Enter decimal data into register.

Figure 6-2. DEBUG Directives (Sheet 1 of 2)

<u>Directive</u>	<u>Description</u>
<u>EFREG</u>	Enter floating point data into register.
<u>END</u>	Terminate execution of both DEBUG and user program.
<u>EREG</u>	Enter half-word hexadecimal data into a register.
<u>EXECUTE</u>	Begin execution of user program at a specified location.
<u>FLOAT</u>	Enter data in floating point.
<u>HEX</u>	Enter half-word hexadecimal data.
<u>IDISPLAY</u>	Display the data contained at the address found at the specified location.
<u>IDREG</u>	Display the data found at the address specified in the given register.
<u>MBKPT</u> or <u>MBKPTR</u>	Set or remove a memory reference breakpoint.
<u>RESTORE</u>	Restore original contents of the system shared library.
<u>ROLL</u>	Display the data following the last display location.
<u>SNAP</u>	Dump to an output file.
<u>STAT</u>	Provide status information such as breakpoints set, last routine referenced, and last directive issued.
<u>STEP</u>	Step through execution of user code one or more instructions at a time.
<u>TRACE</u>	Display traceback information.

Figure 6-2. DEBUG Directives (Sheet 2 of 2)

The following are examples of DEBUG directives:

DI SUBR=500+4,5

If this is the first directive entered under DEBUG or if the last type referenced (if referenced at all) was S, this directive displays five words, starting at four words after location 500 in module SUBR. If SUBR does not contain a label 500, this message is displayed: NO SUCH SYMBOL.

DI 0=(14)/x+((15))/W

Register 14 contains the base address of an array. Register 15 contains the address of an index into the array. This directive displays the location indicated by that index.

HEX SUBR=500/X 1000C 880

Enters two half-words of hexadecimal data at bit address 500 in module SUBR.

BKPT 111/L

Sets a breakpoint at the location corresponding to source line number 111 in the current module. An error message is displayed if the current module is not at least 111 lines long.

DE/H 4A0/ X-10

Places -10 (in decimal form) in the half-word at 4A0 from the beginning of the current module.

DI 0=C840/X

Displays the word with the absolute virtual address of C840.

C #F

Continues through 15 (#F) breakpoints before returning control to you.

Dump or Display Directives

Enter one of the following to display the contents of up to 16 words of virtual memory.

DISPLAY,[name=][location][/type][+offset/type][,nwords]

Displays nwords of hexadecimal and ASCII data.

DDECIMAL[/H],[name=][location][/type][+offset/type][,nwords]

Displays nwords of hexadecimal and decimal data.

DFLOAT[/H],[name=][location][/type][+offset/type][,nwords]

Displays nwords of hexadecimal and floating point data.

IDISPLAY,[name=][location][/type][+offset/type][,nwords]

Displays nwords of hexadecimal and ASCII data, starting at the location indicated by the address specified by the location parameter (indirect addressing).

The dump or display directives use the following parameters:

/H

Indicates that the data to be displayed is half-word data.

name

Name of a module, within the file, relative to which the location parameter is a reference; or 0, in which case the location is an absolute virtual address. An equal sign must immediately follow the name and precede the location, without intervening blanks, in the form name=location. Default name when DEBUG is first started is the main program (or the first module loaded, for non-FORTRAN-generated code); otherwise, the default name is the name last referenced. If the last reference was of the form 0=location, the location is assumed to be an absolute address and an associated type of S or L is disallowed.

location

A hexadecimal address, source line number, statement label, simple variable name, descriptor name, array name, or (hexreg), indicating location at which display is to originate or, for IDISPLAY, the location containing the address indicating the location at which display is to originate. When the offset parameter is present, the location parameter indicates a location relative to which display is to originate. If the location parameter is omitted, the directive uses the address and wordcount from the last display or alter memory directive.

If the location parameter is omitted, the directive uses the address and word count from the last display or alter memory directive.

type

One of the following characters indicating the type of location designated.

- S Statement label, simple variable name, descriptor name, or array name (FORTRAN programs only; not applicable to offset parameter)
- L Source line number (FORTRAN programs only; not applicable to offset parameter)
- X Hexadecimal bit address
- W Hexadecimal word address
- P Hexadecimal page address

At the beginning of DEBUG execution, default type for the location parameter is S, default type for location specified as (hexreg) is X; default type for location when name is specified as 0 is X. For offset, default type at the beginning of execution is X. After the first use of type for a given parameter, the default type for all subsequent uses of the parameter is the type that was last referenced.

offset

Hexadecimal number or (hexreg), indicating an upward or downward offset, in words, from the location indicated by the location parameter. A plus sign or minus sign must immediately precede the number. The only types that are valid with offset are X, W, and P.

nwords

Number of words or half-words to be displayed. Default value is 1; maximum allowed value is 16.

If the number has a # prefix (such as #10), DEBUG interprets the number as a hexadecimal value; otherwise, DEBUG interprets the number as a decimal value.

(hexreg)

The hexadecimal number of a full-word register containing either the desired value or a pointer to an indirect chain culminating in the desired value for a location or offset. hexreg must be enclosed in one set of parentheses. Any additional sets or parentheses indicate another level of indirect addressing. Any type specification attached to the parameter applies only to the ultimate value. Intermediate links in a chain must be hexadecimal bit addresses.

When the controllee file for DEBUG is a FORTRAN program that has been compiled without the S option, dynamic space fields, variables in common areas, and variables that are parameters can be displayed and altered using DEBUG directives. Variables in areas declared common can be displayed by referencing them in the module specified or last referenced. Referencing a descriptor associated with the currently allocated dynamic space fields for the breakpointed module and its higher-level modules displays the contents of those fields. (Higher-level modules are those that have led to the call to the breakpointed module and are linked to it through previous stack pointers) Variables that are parameters in the breakpointed module can be displayed by referencing them in the usual way after the prologue of the breakpointed module has been executed and the variables thereby set to their passed values; during the prologue, their values are indeterminate.

The following directives display virtual memory forward or backward from the last display location.

ROLL[,nwords]

Displays area following last location.

BACK[,nwords]

Displays area preceding last location.

nwords

Number of words to be displayed, starting from last location displayed. Default value is the number of words specified by the previous directive; maximum allowed value is 16.

If the number has a # prefix (such as #10), DEBUG interprets the number as a hexadecimal value; otherwise, DEBUG interprets the number as a decimal value.

Register Directives

Issue one of the following to display and alter the contents of the program registers.

DDREG[/H][,hexreg][,nregisters]

Displays the contents of a full-word or half-word register as hexadecimal and decimal.

DFREG[/H][,hexreg][,nregisters]

Displays the contents of a full-word or half-word register as hexadecimal and floating point.

DREG,[/H][,hexreg][,nregisters]

Displays the contents of a fullword or halfword register is hexadecimal.

EDREG,[/H],hexreg,decidata

Enter full-word or half-word decimal data in registers.

EFREG,[/H],hexreg,fltpt

Enter full-word or half-word floating point data in registers.

EREG,[/H],hexreg,hexdata

Enter fullword or halfword hexadecimal data into registers.

LDREG,hexreg[,nwords]

Displays data found at the address that is given in the specified register.

The register directives use the following parameters:

/H

Indicates that the data to be displayed in half-word data.

hexreg

Full-word or half-word hexadecimal register number that contains data to be displayed or into which data is to be entered. If the register number is omitted, the display register directives use the number and word count from the last display or the entered register directive. Default is the last value used.

nregisters

Number of registers to be displayed, starting with hexreg. Default value is 1; maximum value allowed is 16. If hexreg is also omitted, default is the last value used.

If the number has a # prefix (such as #10), DEBUG interprets the number as a hexadecimal value; otherwise, DEBUG interprets the number as a decimal value.

hexdata

Hexadecimal full-word or half-word data to be entered into n consecutive registers, starting with hexreg. Values are right-justified with zero fill.

fltpt

Floating point data to be entered into consecutive full-word or half-word registers, depending on data type parameters, starting at specified register. E or F format can be used.

decidata

Full-word or half-word decimal data to be entered into consecutive full-word or half-word registers, starting at specified register.

nwords

Number of words to be displayed. Default value is 1; maximum value allowed is 16.

If the number has a # prefix (such as #10), DEBUG interprets the number as a hexadecimal value; otherwise, DEBUG interprets the number as a decimal value.

Alter Memory Directives

Memory locations can be altered in either the controllee's region or the system shared library region. There is a system limitation of only one alteration per word in the shared region. For example, two FLOAT/H directives cannot be used to set the two half-words in a word. However, a single FLOAT/H can be used with two floating point values. Also, the total number of words changed plus breakpoints set in the shared region cannot exceed 20.

Enter one of the following to alter virtual memory.

HEX,[name=]location[/type][+offset[/type]][,halfhex]

Enters hexadecimal data.

ASCII,[name=][location[/type][+offset[/type]][, "ASCIIdata"]

Enters an ASCII character string.

DECIMAL[/H],[name=][location[/type][+offset[/type]][,decidata]

Enters decimal data.

FLOAT[/H],[name=][location[/type][+offset[/type]][,fltpt]

Enters floating point data.

The alter memory directives use the following parameters:

/H

Indicates that the data to be displayed is half-word data.

name

Name of a module, within the file, relative to which the location parameter is a reference; or 0, in which case the location is an absolute virtual address. An equal sign must immediately follow the name and precede the location, without intervening blanks, in the form name=location. Default name when DEBUG is first started is the main program (or the first module loaded, for non-FORTRAN-generated code); otherwise, the default name is the name last referenced. If the last reference was of the form 0=location, the location is assumed to be an absolute address and an associated type of S or L is disallowed.

location

A hexadecimal address, source line number, statement label, simple variable name, descriptor name, array name, or (hexreg), indicating location at which data is to be entered. When the offset parameter is present, the location parameter indicates a location relative to which the data is to be entered.

type

One of the following characters indicating the type of location designated.

S	Statement label, simple variable name, descriptor name, or array name (FORTRAN programs only; not applicable to offset parameter)
L	Source line number (FORTRAN programs only; not applicable to offset parameter)
X	Hexadecimal bit address
W	Hexadecimal word address
P	Hexadecimal page address

Default type when DEBUG is first started is S; otherwise, the default type is the type last referenced.

At the beginning of DEBUG execution, default type for the location parameter is S, default type for location specified as (hexreg) is X; default type for location when name is specified as 0 is X. For offset, default type at the beginning of execution is X. After the first use of type for a given parameter, the default type for all subsequent uses of the parameter is the type that was last referenced.

offset

Hexadecimal number or (hexreg), indicating an upward or downward offset, in words, from the location indicated by the location parameter. A plus sign or minus sign must immediately precede the number. The only types that are valid with offset are X, W, and P.

halfhex

Half-word hexadecimal data values to be entered into consecutive half-word memory locations, starting at location specified. Values are right-justified with zero fill.

ASCIIdata

String of ASCII data to be entered into consecutive character locations, starting at the position given by location parameter. The ASCII data string must be enclosed in quotation marks.

decidata

Full- or half-word decimal data to be entered into consecutive full- or half-word memory locations, beginning at the location specified.

fltpt

Floating point data to be entered into consecutive half- or full-word memory locations, depending on data type parameters, starting at location specified. E or F format can be used.

DEBUG limits the number of breakpoints to 8. However, the total number of words changed plus breakpoints set in the shared region cannot exceed 20. Breakpoints are set on half-word boundaries. The system restriction of one alteration per word in the shared region allows only one breakpoint per word in the shared region.

BKPT, [name=]location[/type][+ offset], $\left\{ \begin{array}{l} \text{IF} \\ \text{IFF} \end{array} \right\}$ ([name=]location[/type][+ offset][.op.]
[name=]location[/type][+ offset])

Defines a conditional breakpoint; the condition to be tested must be enclosed in parentheses. The words IF or IFF are optional for integer quantities. If the quantities to be tested are floating-point, IFF must be specified. IFF results in use of floating-point comparisons; otherwise, the operands are evaluated using integer (48 bits) compare. If the specified condition is met, user program execution stops before the instruction located at the breakpoint address is executed. If the condition is not met, execution continues.

Conditional breakpoints can be set on half-word boundaries, but this uses two half-words in the user program. If the conditional breakpoint overlaps part of a full-word instruction in the user program, results can be undefined. When the condition is not met, the user instructions(s) are executed in a debug buffer area, not in the original position in the user program. If the conditional breakpoint is placed at a relative branch, the branch will not execute correctly and the failure might not be apparent. In addition, if the conditional breakpoint is placed so that the second half-word is branched to, an illegal instruction abort will usually result.

For FORTRAN programs, it would be safest to avoid setting conditional breakpoints at CONTINUE, GOTO, and IF statements. Conditional breakpoints should be set at least one full word before any label that is branched to in the routine being debugged. When this is too restrictive, the DISPLAY command can be used to examine the code at the desired breakpoint location and determine whether the location contains a branch or part of a full-word instruction.

BKPTR, [name=]location[/type][+offset]

Removes an instruction breakpoint. If no parameters are given, all breakpoints in the program are removed.

EXECUTE, [name=]location[/type][+offset]

Causes DEBUG to start executing the user program at the location specified. If no parameters are given on the EXECUTE directive, DEBUG starts at the transfer address. Only one EXECUTE directive can be given per DEBUG run, and it must appear before any CONTINUE directive.

CONTINUE [,nbkpts]

Causes DEBUG to continue executing the user program until it encounters the specified number (nbkpts) of instruction breakpoints. The CONTINUE directive can be entered at any time after entry of the EXECUTE directive.

MBKPT [/qual],[name=]location[/type][,*]

Defines a memory reference breakpoint. Execution of the user program stops as a result of a CPU reference to the location specified. Since the breakpoint is achieved via a data flag branch, execution stops at some time after the actual reference (see discussion of the data flag branch in the CYBER 200 Model 205 Hardware Reference Manual). It is possible to hit only a single breakpoint as a result of several very nearly simultaneous references. Only one memory reference breakpoint may be in use at any time, so setting one implies a removal of any previous one. The optional trailing '*' specification is available for batch use only. It allows DEBUG to continue execution after having encountered the memory reference breakpoint and having recorded it in the output file. This way, you can record a chronology of reference to a location in a single batch run.

MBKPTR

Remove the memory reference breakpoint.

(hexreg)

The hexadecimal number of a full-word register containing either the desired value or a pointer to an indirect chain culminating in the desired value for a location or offset. hexreg must be enclosed in one set of parentheses. Any additional sets or parentheses indicate another level of indirect addressing. Any type specification attached to the parameter applies only to the ultimate value. Intermediate links in a chain must be hexadecimal bit addresses.

Restore Memory Directive

Restores original content of the system shared library that resides in the shared space. If the directives HEX/ASCII/DECIMAL/FLOAT were used to alter the shared library, then RESTORE can be used to restore the original contents of the shared library. RESTORE with no parameters restores all system shared library changes except breakpoints. Breakpoints can be removed by BKPTR.

RESTORE,[name=][location][/type][+offset[/type]]

name

Name of a module, within the file, relative to which the location parameter is a reference; or 0, in which case the location is an absolute virtual address. An equal sign must immediately follow the name and precede the location, without intervening blanks, in the form name=location. Default name when DEBUG is first started is the main program (or the first module loaded, for non-FORTRAN-generated code); otherwise, the default name is the name last referenced. If the last reference was of the form 0=location, the location is assumed to be an absolute address and an associated type of S or L is disallowed.

location

A hexadecimal address, source line number, statement label, simple variable name, descriptor name, array name or (hexreg), indicating location at which data is to be entered. When the offset parameter is present, the location parameter indicates a location relative to which the data is to be entered.

type

One of the following characters indicating the type of location designated.

- | | |
|---|---|
| S | Statement label, simple variable name, descriptor name, or array name (FORTRAN programs only; not applicable to offset parameter) |
| L | Source line number (FORTRAN programs only; not applicable to offset parameter) |
| X | Hexadecimal bit address |
| W | Hexadecimal word address |
| P | Hexadecimal page address |

Default type when DEBUG is first started is S; otherwise, the default type is the type last referenced.

offset

Hexadecimal number or (hexreg), indicating an upward or downward offset, in words, from the location indicated by the location parameter. A plus sign or minus sign must immediately precede the number.

Program Control Directives

Issue one of the following to set and remove breakpoints that start and stop program execution, to dump portions of virtual memory to an output file, or to find the status of DEBUG directives issued earlier.

BKPT,[name=]location[/type][±offset]

Defines an instruction breakpoint. User program execution stops before the instruction located at the breakpoint address is executed. Setting breakpoints within the dynamic linker in SHRLIB can produce undefined results.

DEBUG limits the number of breakpoints to 8. However, the total number of words changed plus breakpoints set in the shared region cannot exceed 20. Breakpoints are set on half-word boundaries. The system restriction of one alteration per word in the shared region allows only one breakpoint per word in the shared region.

BKPT,[name=]location[/type][± offset[/type]],

{ IF } [name=]location[/type][± offset[/type]]
{ IFF }

[.op.][name]location[type][[± offset[/type]]]

Defines a conditional breakpoint; the condition to be tested must be enclosed in parentheses. The words IF or IFF are optional for integer quantities. If the quantities to be tested are floating-point, IFF must be specified. IFF results in use of floating-point comparisons; otherwise, the operands are evaluated using integer (48 bits) compare. If the specified condition is met, user program execution stops before the instruction located at the breakpoint address is executed. If the condition is not met, execution continues.

When the condition is not met, the user instructions(s) are executed in a debug buffer area, not in the original position in the user program. If the conditional breakpoint is placed at a relative branch, the branch will not execute correctly and the failure might not be apparent.

BKPTR,[name=]location[/type][±offset[/type]]

Removes an instruction breakpoint. If no parameters are given, all breakpoints in the program are removed.

EXECUTE,[name=]location[/type][±offset[/type]]

Causes DEBUG to start executing the user program at the location specified. If no parameters are given on the EXECUTE directive, DEBUG starts at the transfer address. Only one EXECUTE directive can be given per DEBUG run, and it must appear before any CONTINUE directive.

CONTINUE [,nbkpts]

Causes DEBUG to continue executing the user program until it encounters the specified number (nbkpts) of instruction breakpoints. The CONTINUE directive can be entered at any time after entry of the EXECUTE directive.

MBKPT [/qual],[name=]location[/type][,*]

Defines a memory reference breakpoint. Execution of the user program stops as a result of a CPU reference to the location specified. Since the breakpoint is achieved via a data flag branch, execution stops at some time after the actual reference (see discussion of the data flag branch in the CYBER 200 Model 205 Hardware Reference Manual). It is possible to hit only a single breakpoint as a result of several very nearly simultaneous references. Only one memory reference breakpoint may be in use at any time, so setting one implies a removal of any previous one. The optional trailing '*' specification is available for batch use only. It allows DEBUG to continue execution after having encountered the memory reference breakpoint and having recorded it in the output file. This way, you can record a chronology of reference to a location in a single batch run.

MBKPTR

Remove the memory reference breakpoint.

STEP [,ninstr]

Causes user program execution to continue for ninstr number of instructions from the last breakpoint encountered. Stepping through a dynamic call to a system shared library module can produce undefined results.

END

Terminates both the user program and DEBUG.

SNAP,[name=]location[/type][+offset[/type]][,nwords]

Dumps nwords words of virtual memory, starting from location, to an output file. Output data is in hexadecimal and ASCII.

SNAP,hexreg,R [,nwords]

Dumps the contents of nwords registers, starting with register numbered hexreg. Output is in hexadecimal and ASCII.

The program control directives use the following parameters:

name

Name of a module, within the file, relative to which the location parameter is a reference; or 0, in which case the location is an absolute virtual address. An equals sign must immediately follow the name and precede the location, without intervening blanks, in the form name=location. Default name when DEBUG is first started is the main program (or the first module loaded, for non-FORTRAN-generated code); otherwise, the default name is the name last referenced. If the last reference was of the form 0=location, the location is assumed to be an absolute address and an associated type of S or L is disallowed.

location

A hexadecimal address, source line number, statement label, simple variable name, descriptor name, or array name, indicating location at which data is to be entered. When the offset parameter is present, the location parameter indicates a location relative to which the data is to be entered.

type

One of the following characters indicating the type of location or offset designated.

S	Statement label, simple variable name, descriptor name, or array name (FORTRAN programs only; not applicable to offset parameters)
L	Source line number (FORTRAN programs only; not applicable to offset parameters)
X	Hexadecimal bit address
W	Hexadecimal word address
P	Hexadecimal page address
R	Register number (conditional breakpoint only)
RH	Half-word register number (conditional breakpoint only)
H	Hexadecimal half-word address (conditional breakpoint only)

At the beginning of DEBUG execution, default type for the location parameter is S, default type for location specified as (hexreg) is X; default type for location when name is specified as 0 is X. For offset, default type at the beginning of execution is X. After the first use of type for a given parameter, the default type for all subsequent uses of the parameter is the type that was last referenced.

qual

One of the following mnemonic acronyms denoting the kinds of memory references for the breakpoint.

R	Read
W	Write
RW	Read or write

offset

Hexadecimal number or (hexreg), indicating an upward or downward offset, in words, from the location indicated by the location parameter. A plus sign or minus sign must immediately precede the number. The only types that are valid are X, W, or P.

ninstr

Number of instructions. Default value is 1; maximum value allowed is 16.

If the number has a # prefix (such as #10), DEBUG interprets the number as a hexadecimal value; otherwise, DEBUG interprets the number as a decimal value.

nwords

Number of words or registers. Default value is 1.

If the number has a # prefix (such as #10), DEBUG interprets the number as a hexadecimal value; otherwise, DEBUG interprets the number as a decimal value.

nbkpts

Number of instruction breakpoints DEBUG must encounter before it returns control to you.

The number is interpreted as a decimal number unless it is preceded by the character #, which indicates that it is a hexadecimal number.

When an nbkpts specification greater than 1 is satisfied, DEBUG displays a frequency count for each breakpoint hit. Encountering a memory reference breakpoint clears any unsatisfied nbkpts count. Frequencies of hits for instruction breakpoints are displayed, but a new CONTINUE[,nbkpts] is needed to resume.

Default value is 1; maximum value is 99999.

op

Acronym denoting condition for the conditional breakpoint. Values are:

EQ	Equal	LE	Less than or equal to
NE	Not equal	GT	Greater than
LT	Less than	GE	Greater than or equal to

(hexreg)

The hexadecimal number of a full-word register containing either the desired value or a pointer to an indirect chain culminating in the desired value for a location or offset. hexreg must be enclosed in one set of parentheses. Any additional sets or parentheses indicate another level of indirect addressing. Any type specification attached to the parameter applies only to the ultimate value. Intermediate links in a chain must be hexadecimal bit addresses.

STAT

Produces a listing of the breakpoints set, the last DEBUG and BKPT directive issued, the last routine name or common block referenced, the last type referenced, and the next execution address in the user program; also displays the last module referenced.

TRACE

Displays the current program address and the stack of callers up to the current callee.

LOOK

You can use the LOOK utility to examine statically any mass storage file to which you have access. It cannot be used during program execution.

After you initiate LOOK execution with the LOOK control statement, the utility responds to directions received from directives. The modifications made through LOOK directives persist for the life of the file modified. LOOK remains in execution until an END is received.

LOOK CONTROL STATEMENT

The control statement that initiates execution of LOOK is shown in figure 6-3. The parameter fname must always be the first parameter, but the order of the INPUT and LIST parameters can be reversed.

LOOK,fname,INPUT=iname,LIST=oname/olen/disp.	
fname	Name of the existing mass storage file being modified or examined by LOOK. To change the file, you must have write access permission to the file.
INPUT=iname	Optional. Name of an existing ASCII file containing directives for LOOK. Default is INPUT=INPUT.
LIST=oname/ olen/disp	Optional. For batch mode, describes the file to which all LOOK output is written. For interactive mode, describes the file to which output is written when the PRINT directive is issued.
oname	Name of the file. Default is OUTPUT.
olen	Integral length of oname in 512-word blocks in decimal; must be greater than 0 and less than 1001. Default is 128 512-word blocks.
disp	Disposition of oname, currently PR only. Default is PR.
If oname exists already, it is destroyed before being created.	

Figure 6-3. LOOK Control Statement Format

LOOK DIRECTIVES

All numbers in all LOOK directives, unless otherwise specified, must be hexadecimal. Directives can be concatenated by slashes. For example, V/W/HEX,1000, 10 is valid, and it indicates three directives that LOOK processes consecutively, just as if they had been issued separately in the same order.

When the parameters for a given directive are meaningless, missing, or illegal, the directive is ignored and an error message is issued.

The response format for the directive is as follows:

```
COMMAND=command keyword
output
```

The following are examples of LOOK directives.

```
PRINT/PAGE/HEX,0,8
```

PRINT indicates that all successive output is to be written to the file OUTPUT.
PAGE indicates that all addresses entered subsequently are page addresses. HEX,0,8 causes a dump of eight full words from the beginning of the first page in the file.

```
HEX,,8
```

Display eight words at the beginning of the first page in the file, assuming the previous PAGE directive. The three columns displayed indicate the word address, hexadecimal contents, and contents in ASCII.

```
SEARCH,`/`,,3
```

Search the file for the third appearance of the character /, beginning at the start of the file. The address at which the specified occurrence is found is reported on the output file.

```
BIT/EASCII,C0,
`1111`/HEX,C0,1
```

BIT indicates that all addresses entered subsequently are bit addresses. EASCII places the ASCII character string 1111 at the half-word address C0. The next directive displays the word just entered in hexadecimal and ASCII.

Optional parameters can be omitted, and a default value is assigned. If an embedded parameter is to be omitted, commas must be used to maintain positional integrity; for example, HEX,,2 defaults the beginning address to 0. If the last one or more parameters are to be omitted, the command must end with the last specified parameter. For example, HEX defaults address to 0 and length to 1; HEX,100 defaults length to 1.

The LOOK directives in figure 6-4 are listed in alphabetical order. The subsequent directive descriptions are grouped according to a common function.

<u>Directive</u>	<u>Description</u>
<u>BACK</u>	Display file portion that immediately precedes portion last displayed.
<u>BASE</u>	All addresses in other LOOK directives are to be offset by a specified amount.
<u>BIT</u>	All addresses in other LOOK directives are to be interpreted as bit addresses.
<u>DEC</u>	Dump or display file in full-word decimal format.
<u>DISPLAY</u>	All output is to go to the terminal.
<u>EASCII</u>	Enter ASCII character string in file at specified location.
<u>EDEC</u>	Enter full-word decimal data items into file at specified location.
<u>EFLOAT</u>	Enter full-word floating point data items into file at specified location.
<u>EHDEC</u>	Enter half-word decimal data items into file at specified location.
<u>EHEX</u>	Enter half-word hexadecimal data items into file at specified location.
<u>EHFLOAT</u>	Enter half-word floating point data items into file at specified location.
<u>END</u>	End LOOK.
<u>FLOAT</u>	Dump or display file in full-word floating point format.
<u>HDEC</u>	Dump or display file in hexadecimal and half-word decimal format.
<u>HEX</u>	Dump or display file in hexadecimal and ASCII format.
<u>HFLOAT</u>	Dump or display file in hexadecimal and half-word floating point format.
<u>HSEARCH</u>	Search file for appearance n of a specified hexadecimal value.
<u>IDEC</u>	Dump or display file in full-word decimal format (indirect addressing of file).
<u>IFLOAT</u>	Dump or display file in hexadecimal and half-word floating point format (indirect addressing of file).
<u>IHDEC</u>	Dump or display file in hexadecimal and half-word decimal format (indirect addressing of file).
<u>IHEX</u>	Dump or display file in hexadecimal and ASCII format (indirect addressing of file).
<u>IHFLOAT</u>	Dump or display file in hexadecimal and half-word floating point format (indirect addressing of file).

Figure 6-4. LOOK Directives (Sheet 1 of 2)

<u>Directive</u>	<u>Description</u>
<u>PAGE</u>	All addresses in other LOOK directives are to be interpreted as page addresses.
<u>PATTERN</u>	Enter pattern into portion of file.
<u>PRINT</u>	All output from directives is to go to the output file.
<u>ROLL</u>	Display file portion that immediately follows portion last displayed.
<u>SEARCH</u>	Search file for appearance n of a specified character string.
<u>SEQ</u>	All addresses in other LOOK directives are to be interpreted as sequential from this point on.
<u>VIRTUAL</u>	All addresses in other LOOK directives are to be interpreted as being virtual.
<u>WORD</u>	All addresses in other LOOK directives are to be interpreted as word addresses.

Figure 6-4. LOOK Directives (Sheet 2 of 2)

SEARCH Directive

LOOK searches a file for an occurrence of a specified string of characters when the SEARCH directive is issued as follows:

SEARCH, 'string', [addr], [n]

string	A string of ASCII characters. The string must be enclosed in single quotes.
addr	Position in file at which search for the character string is to begin, designated by a byte address. If addr does not lie on a byte boundary, it is truncated to the nearest byte boundary. Default is 0.
n	Integer constant greater than zero, indicating the occurrence of the character string that is to be selected. Default is 1.

Beginning at addr, the file is searched for n occurrences of string, and the address of the last is returned. If LOOK is in virtual mode, the search goes through the file virtually. If LOOK is in sequential mode, the search goes through the file sequentially.

If no occurrences of the character string are found, the following message is issued.

CHARACTER STRING NOT FOUND

If m, but fewer than n, occurrences are found when the end of the file is reached, the following message is issued.

ONLY m OCCURRENCES WERE FOUND

If the search is successful, the following message is issued.

CHARACTER STRING FOUND AT address

address is the bit address of the first character of the selected string.

HSEARCH Directive

An HSEARCH directive causes LOOK to search the file for a hexadecimal value. The directive can specify a mask indicating the word fields to be searched.

The following is the directive format.

HSEARCH,hexval,[mask],[addr],[n]

hexval	Hexadecimal value to be found (1 to 16 hexadecimal digits).
mask	64-bit mask r presented by 1 to 16 hexadecimal digits. HSEARCH only searches the bits that are set to 1 in the mask. For example, the mask 000000000000000F indicates that only the rightmost four bits of each word are to be searched. Default is a mask of all 1 bits (all bits are searched).
addr	Bit, word, or page address at which the search is to begin. Default is 0.
n	Occurrence of the value to be found (integer constant greater than 0). For example, 2 indicates the second occurrence of the value. 0 is processed as 1. Default is 1.

LOOK examines a full 64-bit word. If the value being searched for is not right-justified, the value specified on the search command must be adjusted accordingly. For example, if you are searching for the hexadecimal value 7B in the second byte from the right of a word, the command would be:

HSEA,7B00,FF00

If LOOK is in virtual mode, it searches the file by virtual address. If LOOK is in sequential mode, it searches the file sequentially.

Beginning at the specified address, LOOK searches the file for n occurrences of the hexadecimal value. It searches until it finds n occurrences or encounters the end of the file.

If LOOK finds n occurrences of the value, it returns the hexadecimal bit address of the last occurrence with the following message:

HEX VALUE FOUND AT address

If LOOK finds no occurrences of the specified value, it returns the following message:

HEX VALUE NOT FOUND

If LOOK finds fewer than n occurrences of the value, it returns the following message:

ONLY m OCCURRENCES WERE FOUND

Disposition of Directive Output

If you are an interactive user, you can select whether output is to be displayed at the terminal or dumped onto the output file that was specified in the LOOK control statement. Initial mode is DISPLAY. The directives that control the disposition of output are as follows:

PRINT

This indicates that from the time of this directive, or until a DISPLAY or END directive is entered, all output is to be written to the output file.

DISPLAY

This indicates that from the time of this directive, or until a PRINT or END directive is entered, all output is to be sent to the terminal.

END

End LOOK.

Display and Dump Directives

The following directives cause dump or display of a specified number of words of the file, starting at a specified location in the file. For interactive users in display mode, the display is one word or half-word per line. On the output file, four words are placed in one line, with duplicate lines being suppressed. The directives are as follows:

DEC,[addr],[len]

Displays/dumps file portion as hexadecimal and full-word decimal data.

FLOAT,[addr],[len]

Displays/dumps file portion as hexadecimal and full-word floating point data.

HDEC,[haddr],[len]

Displays/dumps file portion as hexadecimal and half-word decimal data.

HEX,[haddr],[len]

Displays/dumps file portion as hexadecimal and ASCII data.

HFLOAT,[haddr],[len]

Displays/dumps file portion as hexadecimal and half-word floating point data.

WARNING

LOOK will extend a file if the file is attached with write permission, and the user displays an address past the end-of-file marker.

The display and dump directives use the following parameters:

addr

Position in file at which display/dump is to begin, designated by a word address. If addr is not on a word boundary, it is truncated to the nearest word boundary. Default is 0.

len

The number of words displayed/dumped. If len is not specified, it is taken to be 1.

haddr

Position in file at which display/dump is to begin, designated by a half-word address. If haddr is not on a half-word boundary, it is truncated to the nearest half-word boundary.

Corresponding to each of the preceding directives is another LOOK directive. This directive performs the identical operation, except that the address parameter specified must be the indirect, rather than the direct, address of the position at which display or dump is to begin. The directives are:

IDEC,[addr],[len]

Displays/dumps file portion as hexadecimal and full-word decimal data.

IFLOAT,[addr],[len]

Displays/dumps file portion as hexadecimal and full-word floating point data.

IHDEC,[haddr],[len]

Displays/dumps file portion as hexadecimal and half-word decimal data.

IHEX,[haddr],[len]

Displays/dumps file portion as hexadecimal and ASCII data.

IHFLOAT,[haddr],[len]

Displays/dumps file portion as hexadecimal and half-word floating point data.

The display and dump directives use the following parameters:

addr

Address of word containing position in file at which display/dump is to begin; if addr is not a full-word address, the value of addr is interpreted to be the first word boundary preceding addr. The low 48 bits of the word at addr is the file position at which the display or dump begins; if the 48 bits do not constitute a word address, the display/dump begins on the first word boundary preceding the address. Default is 0.

len

The number of words displayed/dumped. If len is not specified, it is taken from the top 16 bits of the word at addr or haddr. Default is 1.

haddr

Address of word containing position in file at which display/dump is to begin; if haddr is not a full-word address, the value of haddr is interpreted to be the first word boundary preceding haddr. The low 48 bits of the word at haddr is the file position at which the display or dump begins; if the 48 bits do not constitute a half-word address, the display/dump begins on the first half-word boundary preceding the address. Default is 0.

Additional data can be displayed or dumped with either of the following directives:

ROLL

Displays or dumps the file portion that immediately follows the portion last displayed or dumped. The number of words and format are the same as those of the previous display/dump.

BACK

Displays or dumps the file portion that immediately precedes the portion last displayed or dumped. The number of words and format are the same as those of the previous display/dump.

Directives for Entering Values

LOOK offers seven directives for entering values into the file. The directives cause values to be placed in the file, beginning at a particular location. The operation performed is not an insertion; that is, it does not cause expansion of the size of the file but is, instead, a replacement of the current value with another.

Any file on which these operations are performed must have write access. The directives are as follows:

PATTERN,haddr,laddr,data₀[,data₁...,data_n]

Enters half-word data pattern into the file between haddr and laddr inclusive.

EASCII,haddr,'string'

Enters character string, starting at haddr.

EDEC,haddr,data₀[,data₁...,data_n]

Enters full-word decimal data items, starting at haddr.

EFLOAT,addr,data₀[,data₁...,data_n]

Enters full-word floating point data items, starting at addr.

EHDEC,haddr,data₀[,data₁...,data_n]

Enters half-word decimal data items, starting at haddr.

EHEX,haddr,data₀[data₁...,data_n]

Enters half-word hexadecimal data, starting at haddr.

EHFLOAT,haddr,data₀[,data₁...,data_n]

Enters half-word floating point data items, starting at haddr.

The directives for entering values use the following parameters:

haddr

Position in file at which entry is to begin, designated by a hexadecimal half-word address; haddr must be on a half-word boundary.

laddr

Position in file at which the last half word entered is to be placed, designated by a hexadecimal half-word address; laddr must be on a half-word boundary.

data_i

The data to be entered; either a half word or a full word of data, depending on the directive. Also depending on the directive, data_i is a hexadecimal, decimal, or floating point number that can be represented in a half word or full word.

string

A string of ASCII characters. The string must be enclosed in single quotes. If the number of characters in the string is not a multiple of 4, the last half word is blank filled on the right.

addr

Position in file at which entry is to begin, designated by a hexadecimal full-word address; addr must be on a full-word boundary.

Declaration of Directive Address Type

You can indicate whether addresses in LOOK directives specify a quantity of bits, words, or pages by entering one of the following:

- WORD Specifies that all addresses in LOOK directives are to be interpreted as word addresses.
- PAGE Specifies that all addresses in subsequent LOOK directives are to be interpreted as small page addresses.
- BIT Specifies that all addresses in subsequent LOOK directives are to be interpreted as bit addresses.

Enter the following to indicate that all addresses in LOOK directives are to be offset.

BASE,offset

offset

Offset to be added to all addresses. The offset is in bits, words, or pages, depending on the address mode established by BIT, WORD, or PAGE directives.

Initially, BASE is 0.

Enter one of the following directives to indicate whether the virtual file being manipulated is to be accessed as a virtual or physical file.

SEQ Declares that from this point on, or until VIRTUAL or END is entered, all addresses referred to in other directives are sequential addresses. This allows access to the one or two minus pages of a virtual file, which are not virtually addressable.

VIRTUAL Declares that from this point on, or until SEQ or END is entered, all addresses referred to in other LOOK directives are virtual addresses.

For physical files, directives are meaningless and if entered are ignored; physical files are always in SEQ mode. At the beginning of a LOOK run, mode is VIRTUAL (unless the file is physical).

These specifications hold until another of these directives is entered or LOOK ends. The initial address mode is BIT.

DUMP

The DUMP utility produces a drop file and a controllee file dump as well as a controllee load map and controllee cross-reference map. The load map and cross-reference include all dynamic modules referenced. If DUMP is used on a controllee file, only a controllee load and cross-reference map are produced.

The following describes the information listed in a dump if the LO=D option is chosen and a drop file name is specified.

- Program address.
- Last executed instruction, in hexadecimal.
- System error code of the fatal error condition and the address where the error occurred (also returned in word 139 of the first minus page). Codes are defined in volume 2 of this manual.
- Subroutine traceback. If the error occurred in a subroutine, the address of each CALL statement that led to that subroutine is listed.
- List of the open (active) files at the time the error occurred, along with each file's virtual page address and its length in pages.
- Alpha and Beta words if the word preceding the program address contains an exit force instruction (refer to System Messages in volume 2 of this manual).
- Contents of the first minus page, in hexadecimal and ASCII. Duplicate lines are suppressed.
- Contents of the second minus page, in hexadecimal and ASCII.
- Contents of the third minus page, in hexadecimal and ASCII.
- Contents of memory, in hexadecimal and ASCII, from 50 words preceding the program address to 50 words following the address.
- If the error occurred in a subroutine, the register save area for each caller is dumped. Dumps are output in reverse sequence of the CALL occurrences that led to the failing subroutine. Subroutines are always dumped, regardless of whether they are system-supplied or user-supplied.

The format of the DUMP control statement is shown in figure 6-5. It can be issued either interactively or from within a batch control sequence.

NOTE

The batch processor always dumps the task drop file information if the task sets its abort flag and the controllee file has read access permission. It saves the drop file as a private file belonging to the user number executing the task.

DUMP,filename,LIST=listfile,LO=n.

filename Name of drop file or controllee file. This is a required parameter.

LIST=listfile Name of output listing file. The default is OUTPUT.

LO=n List options in which n can be combinations of the following letter:

- D Produce a drop file and controllee file dump. If the filename specified is a controllee file, DUMP does not produce a dump.
- M Produce a load map.
- X Produce a cross-reference map, where X implies M.
- F Produce a register trace back that includes the contents of the location to which each register points. This is the default for the controllers that do not contain any FORTRAN modules. F implies D.

If LO is not specified, the default is LO=D when filename is a drop file for a static controllee, LO=DM when the drop file is for a dynamic controllee, and LO=X when filename is a controllee file. During batch execution, if a controller aborts, the batch processor automatically calls dump with LO=D and LIST=OUTPUT.

Figure 6-5. DUMP Control Statement Format

When a task requires large amounts of system resources or a long time to execute, it should include one or more checkpoint calls. A checkpoint call writes the current state of the task on a file. If the task later terminates abnormally, the file can be used to restart the task at the point in its processing at which the last checkpoint call was processed. Restarting a large task at an intermediate point in its processing saves system resources and time.

The checkpoint call not only saves the current state of the task containing the checkpoint call, it also saves the current state of any tasks at lower levels in the controllee chain. Restarting the task also restarts the tasks at lower levels in the controllee chain.

CHECKPOINTING A TASK

To provide checkpoints for a task, insert one or more calls to the CHKPNT subroutine in the program. The format of a CHKPNT call is shown in figure 7-1.

CALL CHKPNT (lfn, rst, err, erlfn)	
lfn	Name of file to which the checkpoint information is to be written. Must consist of one through eight letters or digits, beginning with a letter, expressed as a Hollerith constant or as a variable, left justified and blank filled.
rst	Variable to which the system returns a response after checkpoint and restart. 0 Checkpoint executed 1 Restart executed
err	Variable to which the system returns a code for any error found in either checkpoint or restart.
erlfn	Variable to which the system returns the name, left-justified and blank filled, of any file in which errors were encountered during checkpoint or restart. The particular error that causes this variable to be set is documented in the err variable. If the error is not associated with a file, blanks are returned in the erlfn variable.

Figure 7-1. CHKPNT Subroutine Format

You must ensure that each checkpoint occurs at a logical breaking point in task execution. Although the CHKPNT routine restores system message control for the checkpoint task and for any tasks at lower levels in its controllee chain, any messages outstanding when a checkpoint occurs are discarded. Therefore, you should ensure that the checkpoint occurs when no messages could be outstanding.

The CHKPNT call specifies the checkpoint file. CHKPNT copies the current state of the task drop file to the checkpoint file. This file is created by the CHKPNT subroutine. If a file with the same name already exists, it is returned and/or purged by CHKPNT.

If lower-level tasks exist in the controllee chain, CHKPNT creates a checkpoint file for each task. It forms the checkpoint file names by modifying the file name specified in the call with an ending digit, 1 through 9. The ending digits correspond to the level of the controllee in the controllee chain.

The checkpoint files CHKPNT creates are temporary files. To save them as permanent files, the program could call Q5DEFINE to save the temporary checkpoint files or the job could include an EXIT statement followed by a DEFINE statement. Execute the EXIT and DEFINE statements if the task terminates abnormally. The controllee file must also be made permanent, since it must be attached when the task is restarted.

When a program contains more than one CHKPNT call, each call could specify the same checkpoint file name. CHKPNT first copies the drop file to an intermediate file. At the end of its processing, it renames the intermediate file to the name on the call and destroys the existing file having that name. Therefore, if the task aborts during checkpoint processing, the previous checkpoint file still exists.

The intermediate checkpoint file name for the task that contains the CHKPNT call is Q5CKP1; intermediate files for lower-level controllees have the names Q5CKP2 through Q5CKP9.

TASK PROCESSING AFTER THE CHKPNT CALL

After inserting a CHKPNT call in a program, you should also insert statements that check the values returned in variables specified on the call. These statements are executed immediately after checkpoint processing and immediately after the task is restarted.

The first check should be of the return state (rst) variable. The variable value indicates whether the statement is being processed after checkpoint processing or after the task is restarted.

If rst value indicates that checkpoint processing has just completed, statements should then check to see whether a checkpoint error code is returned in the error file (err) specified on the call. Figure 7-2 lists the checkpoint error codes. A Q5DEFINE call could also save the temporary checkpoint files as permanent files.

If the rst value indicates that restart processing has just completed, statements should then check to see whether a restart error code is returned in the err variable. Figure 7-2 lists the restart error codes.

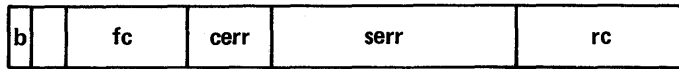
Checkpoint Errors

<u>Code</u>	<u>Meaning</u>
0	No error.
1	Checkpoint file name specified in erlfn duplicates an existing local or attached permanent file name.
2	Input/output connector error.
3	Mass storage or system table space not available.
4	Invalid file name supplied by program.
>4	System error with return word formatted as shown in figure 7-3.
-1	Error in bound implicit map entry for file OUTPUT.

Restart Errors

<u>Code</u>	<u>Meaning</u>
0	No error.
10	Controllee cannot be initialized because system tables are full.
11	More than nine controllee levels.
13	Controllee file specified in erlfn cannot be found.
14	Insufficient time to run controllee specified in erlfn.
15	System error; restart failed.
17	Controllee file specified in erlfn is not executable.
18	Mass storage error for file specified in erlfn.
19	Error in controllee file or drop file input/output connector specified in erlfn.
>19	System error with return word formatted as shown figure 7-3.

Figure 7-2. CHPNT Error Codes



<u>Field</u>	<u>Bits</u>	<u>Meaning</u>
b	0	Busy bit; set if I/O in progress.
	1-3	Unused.
fc	4-15	Function code indicating system function in program (refer to volume 2 of this manual).
cerr	16-23	Central system error code (cerr) for system function.
serr	24-47	Peripheral system error code (serr) for system function.
rc	48-63	Response code for system function.

Figure 7-3. Return Word Format

RESTARTING A TASK

You can restart a checkpointed task by executing the checkpoint file. To execute the checkpoint file, enter a control statement consisting of the name of the checkpoint file. The controllee file must be attached at this point, as must any other files that the task is using.

Restart verifies that all files open at checkpoint time still exist and still occupy the same file space. The file OUTPUT is recreated if necessary.

Executing the checkpoint file system restarts all controllees in the controllee chain of the checkpointed task. Once the controllee chain is reestablished, the system returns control to the checkpointed program at the statement immediately following the CHPNT call. It returns any restart error in the err variable specified on the CHPNT call.

If the restarted task processes another CHPNT call and the checkpoint file specified on the call is the same one used to restart the task, the subsequent checkpoint fails. Therefore, it is recommended that the SWITCH statement be used to change the name of the checkpoint file before it is used to restart the task.

RESTARTING A TASK THAT USES TAPE FILES

Tape files that were open in the checkpointed task are opened when you restart a task. They are not repositioned to reflect their positions at checkpoint.

The following are examples of a job containing a checkpointed task and the job that restarts the checkpointed task:

Job Containing a Checkpointed Task

```
JOB,ST501.
USER,u=69N777,AC=0076A.
RESOURCE,TL=1999,NT=2.
ATTACH,TAPE1,TAPE2
FTN200.
LOAD.
GO.
EXIT.
DEFINE,GO.
DEFINE,CKGO.
End-of-group indicator
  PROGRAM X (INPUT,OUTPUT)
    statements to declare variables
    CALL Q5OPEN(`LFN=`,`TAPE1`)
    CALL Q5OPEN(`LFN=`,`TAPE2`)
    other processing
    CALL CHPNT(`CKGO`,IRST,IERR,IERFL)
    other processing
    CALL Q5CLOSE(`LFN=`,`TAPE1`)
    CALL Q5CLOSE(`LFN=`,`TAPE2`)
    STOP
  END
End-of-information indicator
```

The preceding job saves the checkpoint file, CKGO, only if the job terminates abnormally and the DEFINE statement after the EXIT statement is processed.

Job That Restarts the Checkpointed Task

JOB,ST501.
USER,u=69N777,AC=0076A.
RESOURCE,TL=1000,NT=2.
ATTACH,TAPE1,TAPE2.
ATTACH,CKGO,GO.
CKGO.
End-of-information indicator

The job that restarts the task requests the tapes, using the same control statements that were used to request the tapes in the job that checkpointed the task.

SYSTEM INTERFACE LANGUAGE (NON-I/O CALLS)

8

Chapters 8 and 9 of this manual describe the system interface language (SIL). SIL is a set of subroutines callable by user programs. The user program can be written in FORTRAN, IMPL, or the META assembler language.

Each SIL subroutine formats and issues a system message. The subroutine description lists the system message used. All system messages are described in volume 2 of this manual.

The routines described in chapter 9 of this volume perform functions related to I/O. The routines described in this chapter perform non-I/O functions that enable a task to exchange information with the operating system.

OVERVIEW

Table 8-1 lists the routines described in this chapter, grouped according to a shared function.

Table 8-1. SIL Non-I/O Calls (Sheet 1 of 3)

Inform the system of the task's requirements.

Q5ADVISE	Informs the system of the task's virtual space requirements.
Q5DESBIF	Informs the system that the task should not be rerun if the system fails.
Q5DMPACT	Dumps the cumulative accounting file and terminates its use.
Q5MEMORY	Allocates a static stack.
Q5RECALL	Suspends task execution.
Q5REPREV	Enables or disables user reprieve processing.
Q5RUNBIF	Informs the system that the task should be rerun if the system fails.
Q5SETLP	Changes the current large page limit.
Q5VRACC	Changes the task's accounting rate.

Communicate with the system operator.

Q5GETMOP	Gets a message sent by the system operator.
Q5SNDMOP	Sends a message to the system operator.

Determine task processing if the task encounters an error.

Q5DISATI	Disables abnormal termination control.
Q5ENATI	Enables abnormal termination control.
Q5RFI	Returns control from an interrupt routine.

Determine user reprieve processing.

Q5REPREV	Enables or disables user reprieve processing.
----------	---

Determine message interrupt processing for the task.

Q5DISAMI	Disables message interrupt processing.
Q5ENAMI	Enables message interrupt processing.
Q5RFI	Returns control from an interrupt routine.

Table 8-1. SIL Non-I/O Calls (Sheet 2 of 3)

Initialize and terminate a controllee chain.

Q5INIT	Initializes or initializes and starts a controllee.
Q5INITCH	Initializes a controllee chain.
Q5SNDSTR	Starts controllee execution.
Q5TERM	Terminates a task and its controllee chain.
Q5TERMCE	Disconnects a controllee.

Control message flow within a controllee chain.

Q5GETMCE	Gets a message from a controllee.
Q5GETMCR	Gets a message from a controller.
Q5SNDMCE	Sends a message to a controllee.
Q5SNDMCR	Sends a message to a controller.
Q5SNDMDF	Sends a message to a dayfile.
Q5SNDMJC	Sends a message to the job controller.

Get information about the controllee chain.

Q5GETCTS	Gets the controllee's termination status.
Q5LSTCH	Gets information about each task in the controllee chain.

Get information about the system and the task.

Q5CPUTIM	Gets the CPU time the task has used.
Q5DCDDST	Gets and decodes information from the Disk Status Table.
Q5DCDMSC	Gets and decodes information from the Miscellaneous Table.
Q5DCDPFI	Decodes permanent file indices.
Q5DCDPLB	Decodes a pack label.
Q5GETACT	Gets the system resources the task has used.
Q5GETLP	Gets the task's large page limits.
Q5GETTL	Gets the task's time limit.
Q5GETTN	Gets the task's characteristics.
Q5GETUID	Gets the user number under which the task is running.
Q5TIME	Gets the system time and date.

Table 8-1. SIL Non-I/O Calls (Sheet 3 of 3)

Get permanent file indices to be decoded by Q5DCDPFI.

Q5GETPFI	Copies the label and permanent file indices from a pack.
Q5LFIHIR	Copies attached permanent or local file index entries, depending on a hierarchical search.
Q5LFIPOL	Copies pool file index entries.
Q5LFIPRI	Copies private file index entries.
Q5LFIPUB	Copies public file index entries.

Copy system tables to user-defined arrays.

Q5GETIIP	Copies the interrupted task's invisible package.
Q5GETIRF	Copies the task's register file.
Q5GETMPG	Copies the interrupted task's minus page information.
Q5LSTBUT	Copies the Bank Update Table.
Q5LSTSTB	Copies the statistics buffer.
Q5LSTTCB	Copies the timecard buffer.

SIL ERROR PROCESSING

You can specify three parameters that return information on the last error (if any) encountered during call processing. The three parameters are as follows:

- `STATUS=` ,stat
- `ERRMSG=` ,msg
- `ERRLEN=` ,len

SIL returns the status code of the last error encountered in the variable stat (if specified).

The status code categories are as follows:

<u>Range</u>	<u>Meaning</u>
0	No errors.
1-199	User parameter specification error. The code number is the ordinal of the parameter within the calling sequence.
200-249	Internal error. Consult a systems analyst.
250-9999	Error detected by SIL or the operating system.
10000-11000	Error defined by the site.

Each status code is listed with its associated error message in appendix B of this manual.

If the ERRMSG= parameter is specified, SIL returns an 80-byte error message in the variable specified by the parameter. If the message text is shorter than 80 characters, blanks are appended until the message is 80 characters long. The actual length of the message text is returned in the error message length variable specified by the ERRLEN= parameter.

Each error has a severity level, either fatal or warning. Warning errors return control to the caller. Fatal errors also return control to the caller if the STATUS= parameter is specified on the call. If the STATUS= parameter is omitted from the call and a fatal error occurs, the task is terminated, and a return code of 8 is returned to the controller (refer to Job Termination in chapter 3 of this manual).

Error message routing depends on whether the ERRMSG= parameter is specified and whether the task is to be aborted as a result of the error. The possible actions are summarized as follows:

<u>Action</u>	<u>ERRMSG=,msg Specified</u>	<u>ERRMSG=,msg Omitted</u>
Task to be aborted	Message sent to controller and to msg variable	Message sent to controller
Task not to be aborted	Message sent to msg variable	Message sent to controller

Messages sent to the task's controller usually appear in the dayfile of a batch job or at the terminal of an interactive task.

SIL CALL FORMAT

Calls to SIL routines have the following general format:

```
CALL Q5xxxxxx(p1,p2,...,pn)
```

Q5xxxxxx is the name of an SIL routine. SIL parameters, p_i, have two formats, paired and standalone.

Paired parameters have two parts, a keyword and an option, separated by a comma. The keyword of a paired parameter always ends with an = character. Specify a keyword as a literal or as a variable name containing the keyword. Depending on the parameter, the option may be character data such as an identifier or mnemonic, numeric data such as a buffer or file length, an array address specified by a subscripted variable name, the name of a descriptor, or the name of an external subroutine.

The methods of specifying FORTRAN data formats are described in the CYBER 200 Language Version 1 FORTRAN 200 Reference Manual.

For example, the following call sends a message to the controller. The message is PLEASE ENTER NAME, and it is 12 bytes long. The request status is returned in a variable called STATUS.

```
CALL Q5SNDMCR('MSG=', 'PLEASE ENTER NAME',  
             'LEN=', 12, 'STATUS=', STATUS)
```

Standalone parameters consist of a keyword only. The keyword does not end with an = character. A standalone parameter indicates a logical value (yes or no, on or off) by its presence or omission. Specify the parameter as a literal ('WAIT') or as the name of the variable containing the parameter (refer to the example in the description of No Operation Keywords).

The FORTRAN and IMPL compilers convert the call formats shown in this chapter to the appropriate calling conventions as described in appendix D of volume 2 of this manual. The descriptor of each character parameter (calling or return) must contain a valid length portion specifying the length of the parameter in bytes. SIL considers a length portion of zero as equivalent to a length portion of eight. If the length specified (or assumed) is longer than the actual literal, the literal must be left justified and the remaining characters blanks.

SIL parameters are either required or optional. Required parameters are specified in the individual call formats given later in this chapter. If a required parameter is omitted in a call, SIL returns a fatal error. Certain calls require one parameter specified from a required set. If no parameter is specified from the set, the error message specifies the set number. If an optional parameter is omitted, SIL uses the default value.

Certain calls forbid specification of more than one parameter of a mutually exclusive set of parameters. If more than one parameter is specified from the set, the error message indicates that the second parameter specified from the set is illegal.

When a variable containing character code data is specified, the contents of the variable must be left justified and blank filled to the length specified (or eight bytes). Binary numeric data (such as buffer and file lengths) must be right justified and zero filled.

Keyword and byte (8-bit bytes) parameters begin on byte boundaries unless otherwise stated in the parameter description. Binary numeric data must begin on word boundaries.

NO OPERATION KEYWORDS

SIL recognizes two special no operation keywords, 'NOP=' for paired parameters and 'NOP' for standalone parameters.

The following example illustrates use of the NOP keyword. The following FORTRAN code tests a condition and then either terminates a task abnormally, with an error issued, or normally, with no error issued.

```
EC = 'NOP'  
IF (ERROR) EC = 'ABORT'  
CALL Q5TERM (EC)
```

SIL NON-I/O CALLS

This chapter contains a figure for each SIL routine. The figure contains a call format specifying the required parameters, followed by parameter descriptions.

The parameter descriptions are divided between calling parameters and return parameters. A calling parameter specifies a value used by the SIL routine. A return parameter specifies the name of the variable in which SIL returns a value.

Parameter keywords are listed as FORTRAN literals in the call formats.

Q5ADVISE - ADVISE SYSTEM OF VIRTUAL SPACE REQUIREMENTS

The Q5ADVISE subroutine (refer to figure 8-1) informs the operating system of the task's current virtual space needs. The system uses this information to minimize task paging requirements. It keeps the virtual space that the task indicates it needs paged in, and it pages out the virtual space that the task indicates it no longer needs.

Q5ADVISE recognizes the following two methods of specifying a virtual region:

- Specifying a descriptor word containing the address and length of the virtual region. (Descriptor declaration and initialization are described in the CYBER 200 FORTRAN Language Version 2 Reference Manual.)
- Specifying the name of the first array element in the virtual region. Specify the length of the region. If the length of the region is not specified, Q5ADVISE assumes the default value.

Q5ADVISE uses the Advise system message.

Call Format

```
CALL Q5ADVISE(  { ^INADDR=`,addr  
                  ^INDESC=`,desc } ,optional parameters)  
                  { ^OUTADDR=`,addr  
                  ^OUTDESC=`,desc }
```

Calling Parameters

^INADDR=`,addr	Starting virtual bit address of memory the program needs. If INDESC= is specified, omit INADDR=.
^INDESC=`,desc	Descriptor containing both the length and the address of memory the program needs. The upper 16 bits of the word contain the memory length; the lower 48 bits contain the starting virtual bit address. If INADDR= is specified, omit INDESC=.
^INLEN=`,len	Integer number of memory words the program needs. If INADDR is specified but INLEN= is omitted, SIL assumes that the task needs the 512 words starting at the address specified by INADDR=. If INLEN= and INDESC= are specified, SIL uses the length specified by the INLEN= parameter.
^OUTADDR=`,addr	Starting virtual bit address of the memory the program no longer needs. If OUTDESC= is specified, omit OUTADDR=.
^OUTDESC=`,desc	Descriptor containing both the length and the address of memory the program no longer needs. The upper 16 bits of the word contain the memory length; the lower 48 bits contain the starting virtual bit address. If OUTADDR= is specified, omit OUTDESC=.

Figure 8-1. Q5ADVISE Call Format (Sheet 1 of 2)

Calling Parameters

^OUTLEN=^,len Integer number of memory words the program no longer needs. If OUTADDR= is specified but OUTLEN= is omitted, SIL assumes that the task needs the 512 words starting at the address specified by OUTADDR=. If OUTLEN= and OUTDESC= are specified, SIL uses the length specified by the OUTLEN= parameter.

Return Parameters

^ERRLEN=^,len Error message length in bytes (integer).
^ERRMSG=^,msg 80-byte array to which SIL returns an error message.
^STATUS=^,stat Status code. Possible values: 0 through 202, 250, 450 through 464.

Figure 8-1. Q5ADVISE Call Format (Sheet 2 of 2)

The following examples of FORTRAN source lines illustrate the two methods of specifying a virtual region. In each case, the Q5ADVISE call advises the system that it should keep ARRAY1 paged in, but it can page out ARRAY2.

Virtual region specified by address:

```
DIMENSION ARRAY1(1024), ARRAY2(1024)
CALL Q5ADVISE(^INADDR=^,ARRAY1(1),
+^INLEN=^,1024,^OUTADDR=^,ARRAY2(1),
+^OUTLEN=^,1024)
.
.
.
```

Virtual region specified by descriptor:

```
DIMENSION ARRAY1(1024),ARRAY2(1024)
DESCRIPTOR ADRIN, ADROUT
DATA ADRIN,ADROUT/ARRAY1(1;1024),
+ARRAY2(1;1024)/
CALL Q5ADVISE(^INDESC=^,ADRIN,
+^OUTDESC=^,ADROUT)
.
.
.
```

Q5CPUTIM - GET CPU TIME

The Q5CPUTIM subroutine (refer to figure 8-2) gets the CPU time (in microseconds) that the task has used. A task can issue this call ten times (unless the site changes the installation parameter setting the limit).

Q5CPUTIM uses the Miscellaneous system message.

<u>Call Format</u>	
CALL Q5CPUTIM(^TIME=^,time,optional parameters)	
<u>Calling Parameters</u>	
None.	
<u>Return Parameters</u>	
^ERRLEN=^,len	Error message length in bytes (integer).
^ERRMSG=^,msg	80-byte array to which SIL returns an error message.^STATUS=^,statStatus code. Possible values: 0 through 202, 250, 375.
^TIME=^,time	Task CPU execution time in microseconds (integer). The variable must be a full word on a word boundary. It is a required parameter.

Figure 8-2. Q5CPUTIM Call Format

Q5DCDDST - DECODE DISK STATUS TABLE

The Q5DCDDST subroutine (refer to figure 8-3) copies the disk status table into a buffer it defines, and it retrieves information from the table copy.

SIL returns the information specified by the return parameters on the call. Specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=).

Call Format

CALL Q5DCDDST (one or more parameters)

NOTE

Specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=).

Calling Parameters

`^ENTRY=^,n` Entry number (1 through 128) from which information is retrieved. If ENTRY= is omitted, the entry following the entry specified in the last Q5DCDDST call is used. If this call is the first Q5DCDDST call in the task, the information is retrieved from the first entry in the table.

Return Parameters

`^PNUM=^,num` Pack number (last two characters of the pack name); integer value in the range 1 to 128.

`^LABEL=^,addr` Address (block number) of the disk label; integer value in the range #140 to #9B0.

`^DAU=^,dev` Device allocation unit; integer value in the range 4 to 180.

`^DSNUM=^,dsnum` Device set number (last two characters of the device set name); integer value in the range 1 to 128.

`^DSTDROP=^,ddsnum` Flag indicating whether the set to which the device belongs is available for drop file allocation; integer value.

0 Not a drop file device

1 Available for drop file allocation

`^DSTPU1=^,unit` Primary physical unit number; integer value.

`^DSTZIP1=^,zip` Primary zip code; integer value.

`^ERRS=^,errs` Number of fatal errors from AUTOLOAD to first call of Q5DCDDST in the current task.

Figure 8-3. Q5DCDDST Call Format (Sheet 1 of 2)

Return Parameters

^DTYPE=^,dtype	Device type; integer value in the range 1 to 5 with the following meanings:
	1 Reserved
	2 81912 (18-sector, single density 819)
	3 81922 (18-sector, double density 819)
^DSTDVNO=^,dnum	Device number; integer value.
^DSTPRI=^,flg	Public pack flag.
	0 Public pack
	1 Not a public pack
^DSTUP=^,up	Flag indicating whether pack is logically up.
	0 Pack not up
	1 Pack up
^ERRLEN=^,len	Error message length in bytes (integer).
^ERRMSG=^,msg	80-byte array in which SIL returns an error message.
^NPACK=^,n	Number of entries in the disk status table.
^STATUS=^,stat	Status code. Possible values: 0 through 202, 250, 505.

Figure 8-3. Q5DCDDST Call Format (Sheet 2 of 2)

Q5DCDMSC - DECODE MISCELLANEOUS TABLE

The Q5DCDMSC subroutine (refer to figure 8-4) copies the miscellaneous table into a buffer it defines, and then it retrieves information from the table copy.

SIL returns the information specified by the return parameters on the call. Specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=).

Call Format

CALL Q5DCDMSC (one or more parameters)

NOTE

Specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=).

Return Parameters

'A=',alt	Alternator currently running.
'AA=',alt	Alternator for which the virtual system is running.
'ATIME=',time	Time in the format hh.mm.ss (ASCII characters). The variable must begin on a word boundary.
'CALDLAY=',flg	Write history delay flag.
'CEUSRNO=',un	User number (in ASCII) that can run on-line diagnostics. The variable must begin on a word boundary.
'DATE=',date	Date in ASCII characters. The variable must begin on a word boundary.
'DBLOCK=',db	Descriptor block number for whom all pages are locked.
'ERRLEN=',len	Error message length in bytes (integer).

Figure 8-4. Q5DCDMSC Call Format (Sheet 1 of 5)

Return Parameters

^ERRMSG=^,msg	80-byte array in which SIL returns an error message.
^HILGBLK=^,n	Number of large pages minus one.
^HISTBET=^,b	Beta word for history write.
^HISTEXT=^,flg	Record history flag.
^HISTOFF=^,off	History pointer offset.
^HISTSAV=^,adr	Starting sector address of the history file.
^HISTSIZ=^,len	Length of history file.
^HISTWRT=^,flg	Outstanding write history flag.
^HISTZIP=^,zip	Zip code of the station containing the history file.
^IQMN=^,iqm	User number of the input queue manager (integer).
^LASTALT=^,alt	Standby alternator.
^LPP=^,n	Default lines per page (IP_LPP value).
^MACHID=^,id	Machine identifier.
^MACHINE=^,flg	Hardware type flag. The possible integer values returned are as follows: 5 CYBER 200 Model 205
^MARKER=^,flg	Indicator written in the pack label to indicate pack usage at a particular autoload.
^MAXPAGE=^,n	Maximum available page number.
^MCLOCK=^,clok	ASCII value of the master clock (Wyman clock). The value is in the format yymmddhhsspppp where yy is the year, mm is the month, dd is the day, hh is the hour, ss is the second, and pppp is the decimal fraction of a second.
^MILLSEC=^,sec	Elapsed milliseconds since an arbitrary time base.
^MLPG=^,n	Large page limit for the overall machine.

Figure 8-4. Q5DCDMSC Call Format (Sheet 2 of 5)

Return Parameters

^MXBYCNT=^,mb	Maximum bypass count for a job in the input queue. Refer to Job Scheduling in chapter 3 of this manual for more information.
^MXMO=^,mx	Integer indicating the percentage of memory overcommitment allowed. This value is used when determining whether a job can be scheduled.
^MXRR=^,mxr	Maximum rerun time in system seconds. The combined time limits of all executing jobs cannot exceed this value.
^NOPAGE=^,n	Number of pages in page table.
^NPIPES=^,n	Number of floating-point pipelines in the mainframe. 0 One- or two-pipe machine 1 Four-pipe machine
^OCCPA=^,n	Number of occupied small pages minus one.
^OPRID=^,un	Operator user number; the value is hexadecimal, right-justified, zero-filled.
^OPTB2=^,op	Copy virtual system option.
^OPTCKSM=^,op	Checksumming option.
^OPTHBIT=^,op	Record history option.
^OPTLKVS=^,op	Virtual system lock option.
^OPTNODM=^,op	No drum option.
^OPTSPRG=^,op	Special dedicated memory region option.
^OPTTSBI=^,op	Time stamp option.
^OPTTYOT=^,n	Operator output TTY number.
^OPZIP=^,zip	Zip code of the first-level station used to communicate with the AUTOCON module. The value is hexadecimal and right-justified.
^OSVERS=^,os	Version of operating system that is in use. 0 Version 2.0 or earlier 1 Version 2.1 2 Version 2.2 3 Version 2.3 or later

Figure 8-4. Q5DCDMSC Call Format (Sheet 3 of 5)

Return Parameters

^P=`,db Descriptor block number currently running the alternator.

^PP=`,db Descriptor block number for alternator AA.

^PFADDR=`,adr Physical disk address of paging file.

^PFLEN=`,len Length in small pages of paging file.

^PFUNIT=`,unt Physical unit of paging file.

^PFZIP=`,zip Zip code of paging file.

^RECOVFG=`,n Number of recoveries since last deadstart.

^RERUN=`,rer Batch job rerun flag.

 Y Batch input files are rerun.

 N Batch input files are destroyed at autoloading.

^RSRTFN=`,rsrtinfo File information from which the checkpointed system was restarted.
If RESTART=NO or RESTART=RECOVER was specified for the current
autoloading, then rsrtinfo is zero.

^RVER=`,ver Version number of the resident system.

^SDFCOM=`,flg Comments sent to system dayfile flag (integer).

 0 Comments do not go to the system dayfile.

 1 Comments do go to the system dayfile.

^SHPAR=`,cnt Shift count for large pages.

^SLWS=`,slws Current working set size in blocks for the shared library; zero if
shared libraries are not active.

^SLWSN=`,filename Retrieve the shared library filename.

^SLWSO=`,owner Retrieve the name of the shared library owner (user name in binary,
pool name in ASCII, or 0 for public.)

^SMBKS=`,n Current small page size in 512-word blocks (integer, 1, 4, or 16).

^SMPLG=`,n Number of blocks per large page (always 128).

^STATUS=`,stat Status code. Possible values: 0 through 202, 250, 505.

^SYSID=`,id If the current system uses a system pool, the system pool name is
returned; otherwise, the system identifier is returned (one to eight
characters, left-justified, blank filled).

Figure 8-4. Q5DCMSC Call Format (Sheet 4 of 5)

Return Parameters

^SYSPool=`,n System pool flag (left-justified, blank-filled).
 Y System pool used in the current system. SYSID returns
 the system pool name.
 N System pool not used in the current system.

^SYSUSER=`,sysun Binary value of the system user number.

^TIME=`,clk Microsecond clock.

^TlsBU=`,tl Units in which the job time limit is measured. SIL returns one
 of the following binary values:
 0 System seconds
 1 SBUs

^VPTI=`,vpt Virtual process table index.

^VSSADDR=`,adr Physical disk address of virtual system recovery file.

^VSSLEN=`,len Length of virtual system recovery file.

^VSSUNIT=`,unt Unit number for virtual system recovery file.

^VSSZIP=`,zip Zip code of station for virtual system recovery file.

^VVER=`,ver Version number of virtual system.

^WSZIP=`,zip Workstation zip code.†

†The necessary software and hardware to support a workstation connection are products of
ETA Systems, Inc.

Figure 8-4. Q5DCDMSC Call Format (Sheet 5 of 5)

Q5DCDPFI - DECODE PACK FILE INDEX

The Q5DCDPFI subroutine (refer to figure 8-5) retrieves information from a pack file index entry. Call the Q5GETPFI, Q5LFIPRI, Q5LFIPUB, or Q5LFIPOL subroutine to get a copy of the file index entry before issuing the Q5DCDPFI call.

SIL returns PFI information according to the parameters specified on the Q5DCDPFI call. Specify at least one return parameter (other than ERRLEN=, ERRMSG=, and STATUS=).

The return parameters specified must be appropriate for the device type of the file index entry. If the Q5DCDPFI call specifies mass storage or tape parameters for a file connected to a terminal, Q5DCDPFI returns a warning error (status code 263).

Call Format

CALL Q5DCDPFI(optional parameters)

NOTE

Specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=).

Calling Parameters

- 'ENTRY=',n Relative entry number (beginning with 1) of the file index entry to be decoded. If ENTRY= is omitted, SIL uses the entry number specified on the previous Q5DCDPFI call plus one. If the task has not previously called the Q5DCDPFI routine, the default is 1. The ENTRY= parameter and the MYFILE= parameter are mutually exclusive.
- 'MYFILE=',ary Sixteen-word array containing a user-supplied file index entry. The array must begin on a word boundary. If MYFILE= is omitted, SIL decodes the file index entry obtained by the last Q5GETPFI, Q5LFIPOL, Q5LFIPRI, or Q5LFIPUB call and specified by the ENTRY= parameter. The MYFILE= parameter and the ENTRY= parameter are mutually exclusive.
- 'MYLEN=',len Length of the array specified by the MYFILE= parameter. MYLEN= is required if MYFILE= is specified. The length specified must be 16.
- 'STLEN=',stlen Segment table length used with the SEGADR= and SEGLEN= parameters. The length specified must be 4.

Return Parameters

- 'ACS=',acs Access permission set (left-justified, blank filled).
- | <u>If the file is</u> | <u>Q5DCDPFI returns</u> |
|-----------------------|----------------------------------|
| Private file | Access permission set of caller. |
| Pool file | Pool boss access permission set. |
| Public file | General access permission set. |

Figure 8-5. Q5DCDPFI Call Format (Sheet 1 of 11)

Return Parameters

ACS=,acs returns a one- to five-character string. If no access is allowed, the string is NONE. Otherwise, the string is composed of the following letters:

- R Read permission
- W Write permission
- A Append permission
- M Modify permission
- X Execute permission

^APF=`,x

Access permission flag.

- 0 File does not have a file index table entry extension.
- 1 File has a file index table entry extension.

^ATT=`,cnt

Attached count (integer). For private files, the value returned is the number of users that have the file attached or privileged open; for pool and public files, the value is the number of users that have the file open.

^AU=`,blocks

Allocation unit. The number of 512-word blocks that are allocated when the file is extended. Blocks is an integer value in the range of 1 to 65,535.

^BT=`,bt

Blocking type (ASCII, left justified, blank filled).

<u>blank</u>	<u>Non-SIL file</u>
C	Character count blocking
I	Internal blocking
K	Record count blocking

^DOLA=`,dat

Date of last access to the file returned in the following format (right justified, zero filled). An access is defined as an open but not an attach. For more information, refer to the FILEI description in volume 2 of this manual.

	yy	ddd
--	----	-----

Figure 8-5. Q5DCDPFI Call Format (Sheet 2 of 11)

Return Parameters

<u>Field</u>	<u>Bits</u>	<u>Content</u>
	0-47	Unused
yy	48-54	Last two digits of the year
ddd	55-63	Number of days since the beginning of the year, 1 through 366
<code>^DOLM=^,dat</code>		Date of last open request to this file with write access (right justified, zero filled). The date is returned in the same format as the date returned by the DOLA=parameter. For more information, refer to the FILEI description in volume 2 of this manual.
<code>^DORG=^,dat</code>		Date this file was originated (right justified, zero filled). The date is returned in the same format as the date returned by the DOLA= parameter. For more information, refer to the FILEI description in volume 2 of this manual.
<code>^DT=^,dt</code>		File device. SIL returns one of the following ASCII values (left justified, blank filled). MS Mass storage NT Magnetic tape TE Interactive terminal
<code>^DUP=^,dup</code>		Duplicate file name flag. It is set to #D if a duplicate file entry exists; otherwise, it is set to 1.
<code>^ERRLEN=^,len</code>		Length of the error message in bytes (integer).
<code>^ERRMSG=^,msg</code>		80-byte array to which SIL returns an error message.
<code>^FACT=^,acct</code>		ASCII account number (8 bytes, left justified, blank filled).
<code>^FC=^,fc</code>		File category. SIL returns one of the following ASCII values (left justified, blank filled): B Batch input file N Not defined S System-generated drop file U User file D User-generated drop file

Figure 8-5. Q5DCDPFI Call Format (Sheet 3 of 11)

Return Parameters

^FG=`,fg File acquisition method. SIL returns one of the following ASCII values (left-justified, blank filled):

- Y The user created the file.
- N The user was given the file.

^FI=`,fi Privileged task flag. SIL returns one of the following ASCII values (left-justified, blank filled):

- Y Task is privileged.
- N Task is not privileged.

^LFN=`,lfn File name (ASCII, 8 bytes, left-justified).

^LOCAL=`,loc Indicates that the file is local or permanent. SIL returns one of the following ASCII values (left-justified, blank filled):

- N Not local (permanent)
- Y Local

^MNR=`,mnr Minimum record length in bytes.

^MPN=`,mpn A master project number consisting of one to three alphanumeric characters.

^MXR=`,mxr Maximum record length in bytes.

^OSTAT=`,ostat Output file family status code. Possible values 1 through 31:

ostat	Status
0	Normal status.
1	Destination LID disabled.
2	Destination not responding.
3	Destination rejecting file.
4	SIL error occurred during file transfer.
5	Diverted.
6	Hardware path to LID not available.
7	System error occurred during file transfer.
8	RHF error occurred during file transfer.
9	RWF error occurred during file transfer.
10-27	Reserved by Control Data.
28-31	Reserved for installations.

Figure 8-5. Q5DCDPFI Call Format (Sheet 4 of 11)

Return Parameters

<code>^PC=`,pc</code>	Padding character (ASCII, 1 byte, left-justified).
<code>^PRODTN=`,prod</code>	Indicator of whether a file is a production file or not. SIL returns one of the following ASCII values (left-justified, blank-filled): Y File is a production file. N File is not a production file.
<code>^PTRPFIL=`,pfi</code>	Pointer into pack file index (PFI) for this entry relative to the first page of the PFI. It is set to #FFFF for local files.
<code>^PURGE=`,purge</code>	Indicator of whether the file has been flagged as purge only. SIL returns one of the following ASCII values (left-justified, blank-filled): Y File has purge-only access. N File does not have purge-only access.
<code>^RCT=`,rct</code>	Front-end communication type. SIL returns one of the following ASCII values (left-justified, blank-filled): RHF Remote Host Facility NRHF No Remote Host Facility RWF Remote Workstation Facility
<code>^REF=`,n</code>	Number of times the file has been opened or executed.
<code>^RMK=`,rmk</code>	Record mark character (ASCII, 1 byte, left-justified).
<code>^RP=`,rp</code>	Retention period of the file in days (integer).
<code>^RT=`,rt</code>	Record type. SIL returns one of the following ASCII values: B System block F ANSI fixed length L CYBER Record Manager control word R Record mark delimited U Undefined structure W Control word delimited

Figure 8-5. Q5DCDPFI Call Format (Sheet 5 of 11)

Return Parameters

<code>^SFO=`,sfo</code>	SIL file organization. SIL can return the following ASCII values (left-justified, blank-filled): D Direct access S Sequential access
<code>^SLEV=`,sl</code>	Security level. SIL returns an integer from one through eight.
<code>^ST=`,st</code>	Site identifier. (ASCII, 3 bytes, left-justified). The installation determines the site identifiers. Refer to the FILEI description in volume 2 of this manual for more information.
<code>^STATUS=`,stat</code>	Status code. SIL returns one of the following values: 0 through 199, 250, 261, 262, 263, 504, 505, 506, 508.
<code>^TLR=`,tlr</code>	Time of last open request. SIL returns an integer representing the system clock time, in seconds since midnight, at which the file was last opened.
<code>^TOLM=`,time</code>	Time of last write access. SIL returns an integer representing the system clock time, in seconds since midnight, at which the file was last opened for write access.
<code>^TORG=`,time</code>	Time of file creation. SIL returns an integer representing the system clock time, in seconds since midnight, at which the file was created.
<code>^TYPE=`,typ</code>	File type. SIL returns one of the following ASCII values (left-justified, blank-filled): PD Physical data file VC Virtual code file
<code>^USER=`,un</code>	User number (binary).
<code>^VRI=`,vri</code>	Index into variable rate accounting table (applies to virtual code files only).
<code>^WLEN=`,len</code>	File length in 512-word blocks.
<code>^XCL=`,x</code>	Indicates whether the file can be shared. 0 No task has exclusive use of the file. 1 A task has exclusive use of the file.
<code>^ZIP=`,zip</code>	Zip code for the site identifier specified by the ST parameter.

Figure 8-5. Q5DCDPFI Call Format (Sheet 6 of 11)

Return Parameters for Mass Storage Files Only

<code>^ACT=^,act</code>	Number of active I/O connectors for the file.
<code>^ATJDN=^,jdn</code>	The job descriptor number to which the file is attached (right-justified, zero-filled). The possible integer values are 1 through 2047. If multiple jdns have the file attached, multiple returns are made (one per top).
<code>^CKJDN=^,jdn</code>	The job descriptor number of the checkpointed file (right-justified, zero-filled). The possible integer values are 1 through 999.
<code>^CM=^,cc</code>	Code conversion to be performed at the access station. SIL can return one of the following ASCII values (left-justified, blank-filled): BI Binary. DI Display code (64-character set). EC Extended display code (128-character set).
<code>^CONT=^,con</code>	File contiguity as set when the file was created. SIL returns one of the following ASCII values (left-justified, blank-filled): Y File is contiguous. N File is not contiguous.
<code>^DC=^,dc</code>	Disposition code. If DC= is omitted, SIL assumes that the file is a scratch file. SIL returns one of the following ASCII values (left-justified, blank-filled): IN Input for batch processing. LR Print on a 580-12 line printer. LS Print on a 580-16 line printer. LT Print on a 580-20 line printer. NONE No disposition code set. PR Print on any available printer. PU Punch. P1 Print on a 501 line printer. P2 Print on a 512 line printer. SC Scratch file; destroy at task termination.
<code>^DI=^,di</code>	Job name (eight 6-bit display code characters, right-justified, zero-filled).

Figure 8-5. Q5DCDPFI Call Format (Sheet 7 of 11)

Return Parameters for Mass Storage Files Only

<code>^DUMP=`,dmp</code>	Dump flag. Integer value flag indicates whether the file has been dumped since it was created or last modified. 0 File has not been dumped 1 File has been dumped
<code>^EC=`,ec</code>	Print or punch representation of the file. SIL returns one of the following ASCII values (left-justified, blank-filled): 26 026 punch format 29 029 punch format 80 80-column binary punch format B4 BCD 48-character set B6 BCD 64-character set A4 ASCII 48-character set A6 ASCII 64-character set A9 ASCII 95-character set NONE No external characteristics set
<code>^EXT=`,ext</code>	Extension permission flag as set when the file was created. SIL can return one of the following ASCII values (left-justified, blank-filled): Y File is extendable N File is not extendable
<code>^FIJDN=`,jdn</code>	Job descriptor number (binary). Upon return, jdn is set to the FIJDN field of FILEI and may have integer values from 1 through 2047.
<code>^FISTID=`,id</code>	Terminal identifier (two 6-bit display code characters, right-justified, zero-filled). For more information, refer to the FILEI description in volume 2 of this manual.
<code>^FO=`,fo</code>	File organization. SIL returns one of the following ASCII values (left-justified, blank-filled): B Unstructured binary S Non-SIL structured U Unstructured ASCII

Figure 8-5. Q5DCDPFI Call Format (Sheet 8 of 11)

Return Parameters for Mass Storage Files Only

<code>^GACS=^,acs</code>	General access permission set. GACS=,acs returns a one- to five-character string (left-justified, blank-filled). If no access is allowed, the string is NONE. Otherwise, the string is composed of the following letters: R Read permission W Write permission X Execute permission A Append permission M Modify permission
<code>^GIVETU1=^,un</code>	Binary user number of the user who gave this file to the privileged system task.
<code>^HBW=^,ibyte</code>	The highest byte written to the file. ibyte is an integer in the range of 0 to 8 billion.
<code>^IC=^,ic</code>	File format. SIL can return one of the following ASCII values (left-justified, blank-filled): AS 8-bit ASCII; ANSI carriage control BI Binary PA 8-bit ASCII; ASCII carriage control NONE No internal characteristic set
<code>^LODLEN=^,len</code>	Length, in 512-word blocks, of the program's drop file. If the file is not a virtual code file, this field is zero.
<code>^ORGOWNR=^,un</code>	User number of the originator (binary).
<code>^OSVERS=^,os</code>	Returns a value indicating the release system version on which the file was created. 0 Version 2.0 or earlier 1 Version 2.1 2 Version 2.2 or later

Figure 8-5. Q5DCDPFI Call Format (Sheet 9 of 11)

Return Parameters for Mass Storage Files Only

<code>^OT=`,ot</code>	Origin type of a file destined for the access station. SIL returns one of the following ASCII values (left-justified, blank-filled): B Batch origin E Remote batch origin I Interactive origin
<code>^OVFL=`,overflow</code>	Indicates whether one or more segments of the file have overflowed onto another device. <code>overflow</code> is an ASCII character, either Y or N, left-justified, blank-filled, full-word aligned.
<code>^PARTIAL=`,part</code>	Indicates whether the entry being decoded is for a file that has some of its segments unavailable because a device is down or lost because of file truncation. <code>part</code> is an ASCII character, either Y or N, left-justified, blank-filled, full-word aligned.
<code>^PNUM=`,pnum</code>	The pack number on which the file resides. The number is hexadecimal, right-justified, and zero-filled.
<code>^POOL=`,pool</code>	Pool name (ASCII, left-justified, 8 bytes).
<code>^RERUN=`,flg</code>	Rerun flag (applies to batch input files only). SIL returns one of the following ASCII values: Y Rerun batch job N Do not rerun batch job
<code>^SADDR=`,adr</code>	Disk address of the first block of the file.
<code>^SEGADR=`,adr</code>	Four-word array in which SIL returns the file segment addresses, one address per word. The array must begin on a word boundary. If <code>SEGADR</code> is specified, the <code>STLEN</code> parameter is required. Zero values are returned for files created on a 2.2 or later release system.
<code>^SEGLN=`,len</code>	Array in which SIL returns the file segment lengths, one length per word. The array must begin on a word boundary. If <code>SEGLN</code> is specified, the <code>STLEN</code> parameter is required. Zero values are returned for files created on a 2.2 or later release system.
<code>^TID=`,tid</code>	Terminal identifier for the front end (seven 6-bit display code characters, right-justified, zero-filled). Refer to the <code>FILEI</code> description in volume 2 of this manual for more information.

Figure 8-5. Q5DCDPFI Call Format (Sheet 10 of 11)

Return Parameters for Tape Files Only

`^ADO=`,ado` Assembly/disassembly option used for CYBER 170/CYBER 200 tape interchange (ASCII, left-justified, blank-filled). If selected, each 60-bit portion of tape data read is stored in a CYBER 200 64-bit word, with the upper 4 bits being zero.

BI Binary; no assembly/disassembly performed

BW 60 to 64; assembly/disassembly performed

`^CCS=`,cm` Character conversion mode for tape files (ASCII, left-justified, blank-filled).

AS ASCII character set

EB EBCDIC character set

`^CONVERT=`,cvt` Indicates whether tape data is converted to and from character codes (ASCII, left-justified, blank-filled).

Y Character conversion is performed

N Character conversion is not performed

`^DEN=`,den` Tape recording density (ASCII, left-justified, blank-filled).

PE 1600 cpi

GE 6250 cpi

`^LPROC=`,lp` Tape label processing option (ASCII, left-justified, blank-filled).

R Read existing labels (position and verify HDR1 label)

W Write new labels

`^MFN=`,mfn` Multifile set name (eight ASCII characters).

`^MPRU=`,mpru` MPRU size in bytes for V format tapes (integer).

`^REEL=`,reel` Reel number of the current volume within the multivolume file (integer from 1 through 255).

`^RPB=`,rpb` Records per block for K blocked tapes (integer).

`^TF=`,tf` Tape format (ASCII, left-justified, blank-filled).

I Internal

LB Large block

SI Scope internal

V Variable

NV Non-ANSI, variable

Figure 8-5. Q5DCDPFI Call Format (Sheet 11 of 11)

Q5DCDPLB - DECODE PACK LABEL

The Q5DCDPLB subroutine (refer to figure 8-6) retrieves information from a copy of a pack label. Issue a Q5GETPFI call to get a copy of the pack label before issuing the Q5DCDPLB call.

SIL returns the information specified by the return parameters on the call. Specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=).

Call Format

CALL Q5DCDPLB('PN=',pn,optional parameters)

NOTE

Specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=).

Calling Parameters

'PN=',pn Name of the pack from which SIL gets the pack label. This parameter is required.

Return Parameters

'BADSFN=',badsfn Bad sector filename (ASCII, 8 bytes, left-justified).

'CREATE=',dat Creation date (ASCII characters in the format mm.dd.yy).

'DAU=',blocks The disk allocation unit value for the device. blocks is the integer number of 512-word blocks per allocation unit. The value range is from 4 blocks to one cylinder (180 blocks on 819s).

'DSET=',devset The device set name to which this pack belongs. devset is an ASCII string of eight characters, word aligned, left-justified, and blank filled.

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array in which SIL returns an error message.

'EXPIRE=',dat Expiration date (ASCII characters in the format mm.dd.yy).

'LABEL=',adr Disk block address of this label.

'PACKLEN=',len The length (an integer value) of the device in 512-word blocks.

'PFIE=',n Entry number of this entry within the pack file index, counting from zero.

Figure 8-6. Q5DCDPLB Call Format (Sheet 1 of 2)

Calling Parameters

PFIL=`,len	Original length of the pack file index.
PFILOC=`,adr	Disk block address of the first block of the permanent file index.
SERIES=`,s	The ASCII characters, " Δ 1" (2 bytes, binary value #2031).
STATUS=`,stat	Status code. Possible values: 0 through 199, 202, 250, 261, 505.
TYPE=`,typ	Type of disk pack. The possible values returned are 2 (81912) and 3 (81922).
UPDATE=`,dat	Date of the last update of the disk (ASCII characters in the format mm.dd.yy).
VOLN=`,vol	Volume field (ASCII, 8 bytes, left-justified).

Figure 8-6. Q5DCDPLB Call Format (Sheet 2 of 2)

Q5DESBIF - DESTROY BATCH INPUT FILE

The Q5DESBIF subroutine (refer to figure 8-7) requests that the system destroy the specified batch input file if the system fails.

Q5DESBIF uses the Miscellaneous system message.

The following example of FORTRAN source lines requests that the system destroy the batch input file if the system fails. The name of the batch input file is obtained from a copy of its file index entry via calls to Q5LFIPRI and Q5DCDPFI.

```
CHARACTER*8 LFN
.
.
CALL Q5LFIPRI('BATCH', 'ATTACHED')
CALL Q5DCDPFI('LFN=', LFN)
CALL Q5DESBIF('LFN=', LFN)
.
.
```

Call Format

CALL Q5DCDDST(one or more parameters)

Call Format

CALL Q5DESBIF('LFN=', lfn, optional parameters)

Calling Parameters

'LFN=', lfn Name of the batch input file to be destroyed. The name must be left justified, with blank fill in a full word on a word boundary. This is a required parameter. Specify the LFN= parameter on a Q5DCDPFI call to determine the name of the batch input file.

Return Parameters

'ERRLEN=', len Error message length in bytes (integer).

'ERRMSG=', msg 80-byte array to which SIL returns an error message.

'STATUS=', stat Status code. Possible values: 0 through 202, 250, 303, 400.

Figure 8-7. Q5DESBIF Call Format

Q5DISAMI - DISABLE MESSAGE INTERRUPTS

The Q5DISAMI subroutine (refer to figure 8-8) disables message interrupt processing by the task. Message interrupt processing is enabled by a Q5ENAMI call.

Q5DISAMI uses the Program Interrupt system message.

<u>Call Format</u>	
CALL Q5DISAMI(optional parameters)	
<u>Calling Parameters</u>	
None.	
<u>Return Parameters</u>	
^ERRLEN=^,len	Error message length in bytes (integer).
^ERRMSG=^,msg	80-byte array to which SIL returns an error message.
^STATUS=^,stat	Status code. Possible values: 0 through 202, 250.

Figure 8-8. Q5DISAMI Call Format

Q5DISATI - DISABLE ABNORMAL TERMINATION CONTROL

The Q5DISATI subroutine (refer to figure 8-9) disables the abnormal termination control feature described under Abnormal Termination Control in chapter 3 of this manual.

Q5DISATI uses the Abnormal Termination Control system message.

<u>Call Format</u>	
CALL Q5DISATI(optional parameters)	
<u>Calling Parameters</u>	
None.	
<u>Return Parameters</u>	
^ERRLEN=`,len	Error message length in bytes (integer).
^ERRMSG=`,msg	80-byte array to which SIL returns an error message.
^STATUS=`,stat	Status code. Possible values: 0 through 202, 250.

Figure 8-9. Q5DISATI Call Format

Q5DMPACT - DUMP CUMULATIVE ACCOUNTING BUFFER

The Q5DMPACT subroutine (refer to figure 8-10) dumps the cumulative accounting file to permanent storage and terminates the temporary file. Only a privileged user can issue the Q5DMPACT call.

Q5DMPACT uses the Accounting Communication system message.

Call Format

CALL Q5DMPACT(optional parameters)

Calling Parameters

None.

Return Parameters

^ERRLEN=^,len	Error message length in bytes (integer).
^ERRMSG=^,msg	80-byte array to which SIL returns an error message.
^STATUS=^,stat	Status code. Possible values: 0 through 202, 250.

Figure 8-10. Q5DMPACT Call Format

Q5ENAMI - ENABLE MESSAGE INTERRUPTS

The Q5ENAMI subroutine (refer to figure 8-11) enables messages from an interactive terminal to interrupt the task. When the task is interrupted, the specified interrupt subroutine is executed. To return control to the interrupted task, the interrupt subroutine must issue a Q5RFI call.

Q5ENAMI uses the Program Interrupt system message. Message interrupt processing is described in the Program Interrupt system message description in chapter 2 of this manual.

<u>Call Format</u>	
CALL Q5ENAMI(`SUBNAME=`,sub,optional parameters)	
<u>Calling Parameters</u>	
`SUBNAME=`,sub	Name of your interrupt subroutine. The subroutine (declared external in the calling program) is called if a message interrupt occurs. This is a required parameter.
`TERMINAL`	Indicates that you interrupted the program with terminal messages preceded by the left justified characters (sc)I; (sc) is a special character defined by the installation (refer to Interactive Request Lines in chapter 3 of this manual). If `TERMINAL` is omitted, all messages from a terminal interrupt the program.
<u>Return Parameters</u>	
`ERRLEN=`,len	Error message length in bytes (integer).
`ERRMSG=`,msg	80-byte array to which SIL returns an error message.
`STATUS=`,stat	Status code. Possible values: 0 through 202, 250, 380.

Figure 8-11. Q5ENAMI Call Format

The following example of FORTRAN source lines enables message interrupt processing and specifies INTRUPT as the interrupt subroutine. A \$I interactive request line interrupts the task and gives control to the INTRUPT subroutine.

```
EXTERNAL INTRUPT
.
.
.
CALL Q5ENAMI(`SUBNAME=`,INTRUPT,`TERMINAL`)
```

The following example of an interrupt subroutine returns a status message for each interrupt. A status message is assigned by referencing an index variable and an array of character strings in a common block. It is assumed that the main program updates the status index variable.

```
SUBROUTINE INTRUPT
COMMON/COM1/STATINDX, STABLE(5)
INTEGER STATINDX
CHARACTER*80 STATMSG, STABLE
STATMSG = STABLE(STATINDX)
CALL Q5SNDMCR ('MSG=',STATMSG,'LEN=',80)
CALL Q5RFI
RETURN
END
```

Q5ENATI - ENABLE ABNORMAL TERMINATION CONTROL

The Q5ENATI subroutine (refer to figure 8-12) enables the abnormal termination control feature described under Abnormal Termination Control in chapter 3 of this manual.

Q5ENATI uses the Abnormal Termination Control system message.

Call Format

```
CALL Q5ENATI('SUBNAME=',sub,optional parameters)
```

Calling Parameters

'ERRLIM=',lim The maximum number (1 to 256) of error recoveries (excluding time limit errors). When this limit is exceeded, abnormal termination control aborts the task. If lim exceeds 256, the value of its lowest 8 bits is used (no error is recorded). If ERRLIM=',lim is omitted, the default limit is 25 recoveries.

'SUBNAME=',sub The name of your interrupt subroutine or an entry point within your interrupt subroutine. Control transfers to the entry point if a predefined system fatal error occurs. You must declare the subroutine as external in the calling program. This is a required parameter.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 299, 380, 381.

The following example of FORTRAN source lines enables abnormal termination control and specifies ATSUB as the interrupt subroutine.

```
EXTERNAL ATSUB
.
.
.
CALL Q5ENATI('SUBNAME=',ATSUB)
```

Figure 8-12. Q5ENATI Call Format

Q5GETACT - GET RESOURCE USAGE STATISTICS

The Q5GETACT subroutine (refer to figure 8-13) obtains user accounting information from the controllee minus page. Specify at least one return parameter (other than the ERRLEN=, ERRMSG=, and STATUS= parameters).

The task can use the values returned in the SYSCHRG= and USRCHRG= variables to determine whether it is approaching its time limit. The SYSCHRG= and USRCHRG= parameters return the number of system time units (STUs) and system billing units (SBUs) used by the task. Either STUs or SBUs are used by the system to limit resource usage; the unit used depends on a system variable setting.

Q5GETACT uses the User/Accounting Communication system message.

Call Format

CALL Q5GETACT(parameters)

NOTE

Specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=).

Calling Parameters

None.

Return Parameters

^CPUTIME=,tim	User execution CPU time in microseconds.
^CWSSIZ=,siz	Current working size.
^DPLPEXP=,n	Number of large page explicit reads and writes to or from mass storage.
^DPLPFLT=,n	Number of mass storage reads due to large page faults.
^DPLPIMP=,n	Number of large page implicit writes to mass storage.
^DPSPEXP=,n	Number of small page explicit reads and writes to and from mass storage.
^DPSPFLT=,n	Number of mass storage reads due to small page faults.
^DPSPIMP=,n	Number of small page implicit writes to mass storage.
^DPXFEXP=,n	Number of sectors transferred to mass storage for explicit reads and writes.

Figure 8-13. Q5GETACT Call Format (Sheet 1 of 2)

Return Parameters

'DPXFIMP=',n	Number of sectors transferred to mass storage for implicit writes.
'ERRLEN=',	Error message length in bytes.
'ERRMSG=',msg	80-byte array to which SIL returns an error message.
'LLPC=',lpc	Lost large page count; number of large page requests resulting from swapping of the task (integer).
'LSPC=',lpc	Lost small page count; number of small page requests resulting from swapping of the task (integer).
'MEMUSE=',usg	Memory usage. At the end of each account period, the current working set size is multiplied by the user CPU time for the current accounting period and the product is added to a running total kept in this field.
'MTACCES=',access	Number of magnetic tape reads and writes.
'MTNONIO=',nio	Number of non-I/O magnetic tape operations.
'MTXFER=',xfer	Number of 16-bit byte transfers to and/or from magnetic tape.
'SYSTIME=',tim	System CPU execution time used by the task in microseconds.
'STATUS=',stat	Status code. SIL returns one of the following values: 0 through 202, 250, 261.
'SYSACCT=',astu	Number of STUs accumulated for the account block (integer).
'SYSCHRG=',n	Number of STUs used by the task.
'USRACCT=',asbu	Number of SBUs accumulated for the account block (real value).
'USRCHRG=',sbu	SBUs used by the task (real value).
'VSCALL=',n	Number of virtual system user calls made.
'WS2SM=',n	Cumulative CPU time, in microseconds, that this task's working set limit appeared to be too small. The working set limit is specified on the RESOURCE statement or the interactive execute line.

Figure 8-13. Q5GETACT Call Format (Sheet 2 of 2)

Q5GETCTS - GET CONTROLLEE TERMINATION STATUS

The Q5GETCTS subroutine (figure 8-14) gets the termination status of the task's controllee.

Q5GETCTS also returns the system return code for the controllee.

Q5GETCTS uses the Miscellaneous system message.

Call Format

```
CALL Q5GETCTS( { 'CTS=',cts  
                'RETCODE=',ret, } optional parameters)
```

Calling Parameters

None.

Return Parameters

'CTS=',cts	Controllee termination status (ASCII, left-justified, blank filled). The variable must be a full word on a word boundary. CTS= must be specified if RETCODE= is omitted.
AB	Controllee aborted.
OB	Operator transferred control to end-of-job card.
OD	Operator dropped job.
OE	Operator transferred control to EXIT card.
SA	Controllee still active.
TN	Controllee terminated; files not saved.
TS	Controllee terminated; files saved.
UB	User entered terminal message transferring control to EXIT card.
'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	80-byte array to which SIL returns an error message.
'RETCODE=',ret	System return code (ASCII, left-justified, blank filled). The variable must be a full word on a word boundary. RETCODE= must be specified if CTS= is omitted.
blank	No error
ERROR	Warning error (nonfatal)
FATAL	Fatal error
'STATUS=',stat	Status code. Possible values: 0 through 202, 250.

Figure 8-14. Q5GETCTS Call Format

Q5GETIIP - GET INVISIBLE PACKAGE

The Q5GETIIP subroutine (refer to figure 8-15) gets a copy of the task's invisible package after the task has been interrupted. The invisible package contains the address and control information required to continue execution of the task. The format of the invisible package is described in appendix E of volume 2 of this manual.

Q5GETIIP uses the Miscellaneous system message.

<u>Call Format</u>	
CALL Q5GETIIP(‘INVPACK=’,inv,optional parameters)	
<u>Calling Parameters</u>	
None.	
<u>Return Parameters</u>	
‘ERRLEN=’,len	Error message length in bytes (integer).
‘ERRMSG=’,msg	80-byte array to which SIL returns an error message.
‘INVPACK=’,inv	40-word array in which SIL returns the invisible package. This is a required parameter.
‘STATUS=’,stat	Status code. Possible values: 0 through 202, 250, 383.

Figure 8-15. Q5GETIIP Call Format

Q5GETIRF - GET REGISTER FILE

The Q5GETIRF subroutine (refer to figure 8-16) gets a copy of the task's register file after the task has been interrupted. The register file consists of the contents of the 256 CYBER 200 hardware registers when the task is interrupted. The register file is saved when the job is interrupted. The format of the register file is described in appendix D of volume 2 of this manual.

Q5GETIRF uses the Miscellaneous system message.

Call Format

CALL Q5GETIRF(‘REGFILE=’,rf,optional parameters)

Calling Parameters

None.

Return Parameters

‘ERRLEN=’,len	Error message length in bytes (integer).
‘ERRMSG=’,msg	80-byte array to which SIL returns an error message.
‘REGFILE=’,rf	256-word array in which SIL returns the register file. This is a required parameter.
‘STATUS=’,stat	Status code. Possible values: 0 through 202, 250, 383.

Figure 8-16. Q5GETIRF Call Format

Q5GETLP - GET LARGE PAGE LIMITS

The Q5GETLP subroutine (refer to figure 8-17) gets the large page limits for the task. The maximum large page limit is set by the job RESOURCE statement, the task execute line, or an installation-defined default value. The current large page limit is either the maximum large page limit or the limit specified on a previous SET statement or Q5SETLP call.

Q5GETLP uses the Process System Parameter system message.

The following is an example of a call requesting that SIL return the maximum large page limit in variable LPAGES and the current large page limit in variable NPAGES.

```
CALL Q5GETLP(`NLP=`,LPAGES,`RLP=`,NPAGES)
```

<u>Call Format</u>	
CALL Q5GETLP (`NLP=`,nlp,optional parameters)	
<u>Calling Parameters</u>	
None.	
<u>Return Parameters</u>	
`ERRLEN=`,len	Error message length in bytes (integer).
`ERRMSG=`,msg	80-byte array to which SIL returns an error message.
`NLP=`,nlp	Integer variable in which SIL returns the maximum large page limit for the task.
`RLP=`,rlp	Integer variable in which SIL returns the current large page limit for the task.
`STATUS=`,stat	Status code. Possible values: 0 through 202.

Figure 8-17. Q5GETLP Call Format

Q5GETMCE - GET MESSAGE FROM CONTROLLEE

The Q5GETMCE subroutine (refer to figure 8-18) obtains a message from the task's controllee.

Q5GETMCE uses the Get Message From Controllee system message.

Call Format

CALL Q5GETMCE('MSG=',msg,optional parameters)

Calling Parameters

'LEN=',len	Message buffer length in bytes. If LEN= is omitted, SIL assumes an 80-byte message buffer. The maximum possible length is 4096 bytes.
'NULLFILL'	Indicates the fill character used if STD or SYD is specified. If NULLFILL is specified, binary zero is used. Otherwise, blank fill is used. NULLFILL must be omitted if STD and SYD are omitted.
'RJUSTIFY'	Indicates justification of symbols if STD or SYD is specified. If RJUSTIFY is specified, symbols are right-justified. Otherwise, symbols are left-justified. RJUSTIFY must be omitted if STD and SYD are omitted.
'SAVE'	Indicates that the system buffer space is to be saved. If SAVE is omitted, the buffer space is released.
'STD'	Indicates use of standard delimiters (period, blank, comma, slash, equal, plus, minus, left parenthesis, and right parenthesis). STD and SYD are mutually exclusive. If neither is specified, the message is unedited, left-justified, and null filled.
'SYD'	Indicates use of system delimiters (defined by an installation parameter). STD and SYD are mutually exclusive. If neither is specified, the message is unedited, left-justified, and null filled.

Return Parameters

'DB=',db	Descriptor block number of the controllee that sent the message.
'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	80-byte array to which SIL returns an error message.
'LEVEL=',lev	Level, within the controllee chain, of the controllee that sent the message.
'MSG=',msg	Array to receive (optionally edited) message. This is a required parameter.
'MSGLEN=',n	Number of bytes received. The value returned is the number of characters unless STD is specified, in which case it is the number of items (words).
'STATUS=',stat	Status code. Possible values: 0 through 202, 250, 261, 340, 341, 342, 344, 345.

Figure 8-18. Q5GETMCE Call Format

Q5GETMCR - GET MESSAGE FROM CONTROLLER

The Q5GETMCR subroutine (refer to figure 8-19) obtains a message from the task's controller.

If a controllee other than a batch processor or interactive processor controllee (level 3 or greater) issues a Q5GETMCR call when it has no message waiting for it, controllee execution is suspended and its controller executes until it sends a message to the controllee.

Q5GETMCR uses the Get Message From Controller system message.

<u>Call Format</u>	
CALL Q5GETMCR('MSG=',msg,optional parameters)	
<u>Calling Parameters</u>	
'LEN=',len	Message buffer length in bytes. If LEN= is omitted, SIL assumes an 80-byte message buffer. The maximum possible length is 4096 bytes.
'NULLFILL'	Indicates the fill character used if STD or SYD is specified. If NULLFILL is specified, binary zero is used. Otherwise, blank fill is used. NULLFILL must be omitted if STD and SYD are omitted.
'REJECT'	Indicates that SIL should return an error code if a message is not waiting. If REJECT is omitted, SIL suspends task execution until a message is available.
'RJUSTIFY'	Indicates justification of symbols if STD or SYD is specified. If RJUSTIFY is specified, symbols are right-justified. Otherwise, symbols are left-justified. RJUSTIFY must be omitted if STD and SYD are omitted.
'SAVE'	Indicates that the system buffer space is to be saved. If SAVE is omitted, the buffer space is released.
'STD'	Indicates use of standard delimiters (period, blank, comma, slash, equal, plus, minus, left parenthesis, and right parenthesis). STD and SYD are mutually exclusive. If neither is specified, the message is unedited, left-justified, and null filled.
'SYD'	Indicates use of system delimiters (defined by an installation parameter). STD and SYD are mutually exclusive. If neither is specified, the message is unedited, left-justified, and null filled.
'TIME=',sec	Minimum time interval to elapse before control is returned to the task if a message is not available in seconds (30 through 1800). If the value is greater than 1800, SIL sets the value to 1800; if the value is less than 30, SIL sets the value to 30. The elapsed interval may exceed this time by up to 30 seconds. If TIME=,sec is omitted, SIL suspends controllee execution until a message is available.

Figure 8-19. Q5GETMCR Call Format (Sheet 1 of 2)

Return Parameters

DB=,db	Descriptor block number of the controller that sent the message.
ERRLEN=,len	Error message length in bytes (integer).
ERRMSG=,msg	80-byte array to which SIL returns an error message.
LEVEL=,lev	Level within the controllee chain of the controller that sent the message.
MSG=,msg	Array to receive (optionally edited) message. This is a required parameter.
MSGLEN=,n	Number of bytes received (decimal integer). If the number of bytes received is zero and the status code returned is zero, SIL returns control to the task when the specified time interval elapses, and no message is available. The value returned is the number of characters unless STD is specified, in which case it is the number of items (words).
STATUS=,stat	Status code. Possible values: 0 through 202, 250, 261, 340 through 343.

Figure 8-19. Q5GETMCR Call Format (Sheet 2 of 2)

Q5GETMOP - GET MESSAGE FROM OPERATOR

The Q5GETMOP subroutine (refer to figure 8-20) obtains a message from the operator.

Q5GETMOP uses the Get Message From Operator system message.

<u>Call Format</u>	
CALL Q5GETMOP(‘MSG=’,msg,optional parameters)	
<u>Calling Parameters</u>	
‘LEN=’,len	Message buffer length in bytes. If LEN= is omitted, SIL assumes an 80-byte message buffer.
‘NULLFILL’	Indicates the fill character used if STD or SYD is specified. If NULLFILL is specified, binary zero is used. Otherwise, blank fill is used. NULLFILL must be omitted if STD and SYD are omitted.
‘REJECT’	Indicates that SIL should return an error code if a message is not waiting. If REJECT is omitted, SIL suspends task execution until a message is available.
‘RJUSTIFY’	Indicates justification of symbols if STD or SYD is specified. If RJUSTIFY is specified, symbols are right-justified. Otherwise, symbols are left-justified. RJUSTIFY must be omitted if STD and SYD are omitted.
‘STD’	Indicates use of standard delimiters (period, blank, comma, slash, equal, plus, minus, left parenthesis, and right parenthesis). STD and SYD are mutually exclusive. If neither is specified, the message is unedited, left-justified, and blank filled.
‘SYD’	Indicates use of system delimiters (defined by an installation parameter). STD and SYD are mutually exclusive. If neither is specified, the message is unedited, left-justified, and blank filled.
<u>Return Parameters</u>	
‘DB=’,db	Descriptor block number of the operator task.
‘ERRLEN=’,len	Error message length in bytes (integer).
‘ERRMSG=’,msg	80-byte array to which SIL returns an error message.
‘MSG=’,msg	Array to receive (optionally edited) message. This is a required parameter.
‘MSGLEN=’,n	Number of bytes received. The value returned is the number of characters unless STD is specified, in which case it is the number of items (words).
‘STATUS=’,stat	Status code. Possible values: 0 through 202, 250, 261, 340 through 342.

Figure 8-20. Q5GETMOP Call Format

Q5GETMPG - GET MINUS PAGE

The Q5GETMPG subroutine (refer to figure 8-21) gets a copy of the task's first and second minus page information. It does not return the third minus page information even if the third minus page exists. The operating system uses the minus page information during task execution, as described in chapter 2 of volume 2 of this manual.

Q5GETMPG uses the Miscellaneous system message.

Call Format

CALL Q5GETMPG('MPAGE=',mpage,optional parameters)

Calling Parameters

None.

Return Parameters

'BUFLLEN=',len	Length of the MPAGE buffer. The length must be 512, 1024, or 1536 to get first, second, and third minus pages. If this parameter is not specified, the buffer must be 1024.
'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	80-byte array to which SIL returns an error message.
'MPAGE=',mp	1536-word array in which SIL returns information from the task's minus pages. The array must be on a word boundary. Word 513 or 1025 contains the value #FFFF if a second or third minus page is not returned. This is a required parameter.
'STATUS=',stat	Status code. Possible values: 0 through 202, 340, 250.

Figure 8-21. Q5GETMPG Call Format

Q5GETPFI - GET PACK LABEL AND FILE INDEX

The Q5GETPFI subroutine (refer to figure 8-22) gets a copy of the file indices from the specified disk pack. Only a privileged user or a master user of an account identifier can call Q5GETPFI.

SIL copies the file indices into a buffer it defines. All the PFI entries are returned for a privileged user. If you are a master user, only those PFI entries for which you are a master user of the account identifier will be returned for the specified pack. Retrieve information from the file indices with the Q5DCDPFI subroutine.

The buffer used by the Q5GETPFI routine is the same buffer used by the Q5LFIHIR, Q5LFI PRI, Q5LFI POL, and Q5LFI PUB routines. A call to any of these routines overwrites the contents of the buffer.

Because Fortran I/O uses the Q5LFI XXX routines, invoking Fortran I/O will cause the buffer to be overwritten. All Q5DCDPFI processing following a Q5LFI XXX or Q5GETPFI call must be done without any intervening Fortran I/O calls.

Q5GETPFI uses the Get Pack Label and PFI system message for an unformatted PFI.

Call Format

CALL Q5GETPFI(^PN=^,pn,optional parameters)

Calling Parameters

^PN=^,pn Name of the pack from which SIL gets the pack label and file indices. Pack name is 6 ASCII characters, left justified, blank filled, of the form PACKnn where nn is the pack number. This parameter is required.

Return Parameters

^ERRLEN=^,len Error message length in bytes (integer).
^ERRMSG=^,msg 80-byte array to which SIL returns an error message.
^NFILES=^,n Number of file indices returned.
^STATUS=^,stat Status code. Possible values: 0 through 202, 250, 261, 300, 310 through 312.

Figure 8-22. Q5GETPFI Call Format

Q5GETTL - GET TIME LIMIT

The Q5GETTL subroutine (refer to figure 8-23) gets the existing time limit of the task.

Q5GETTL uses the Miscellaneous system message.

<u>Call Format</u>	
CALL Q5GETTL('OLDTIME=',tl,optional parameters)	
<u>Calling Parameters</u>	
None.	
<u>Return Parameters</u>	
'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	80-byte array to which SIL returns an error message.
'OLDTIME=',tl	Existing time limit. This parameter is required.
'STATUS=',st	Status code. Possible values: 0 through 202, 250.

Figure 8-23. Q5GETTL Call Format

Q5GETTN - GET TASK ATTRIBUTES

The Q5GETTN subroutine (refer to figure 8-24) can get the following information about a task:

- Source file name
- Drop file name
- Job descriptor number
- Level in the controllee chain
- Name of associated batch jobs
- User number
- Originating site identifier
- Privileged status

Q5GETTN uses the Miscellaneous system message.

Call Format

CALL Q5GETTN(one or more parameters)

NOTE

Specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=).

Calling Parameters

None.

Return Parameters

ˆBINARY=ˆ,lfn	Source file name for the task, (ASCII, left justified, blank filled).
ˆDROPFIL=ˆ,lfn	Drop file name for the task, (ASCII, left justified, blank filled).
ˆERRLEN=ˆ,len	Length of error message in bytes (integer).
ˆERRMSG=ˆ,msg	80-byte array in which SIL returns an error message.
ˆJOBNAME=ˆ,jobname	Job name (ASCII, left justified, blank filled). (For interactive tasks, all blanks returned.)
ˆLEVEL=ˆ,lev	Level of this task in the controllee chain.
ˆPRIV=ˆ,prv	Privileged user flag. SIL returns one of the following ASCII values (left justified, blank filled):
	Y Privileged
	N Nonprivileged

Figure 8-24. Q5GETTN Call Format (Sheet 1 of 2)

Return Parameters

<code>^ST=^,siteid</code>	Originating site identifier (three-character ASCII, left-justified, blank-filled).
<code>^STATUS=^,stat</code>	Status code. SIL returns one of the following values: 0 through 202, 250, 261.
<code>^JDN=^,jdn</code>	The job descriptor number to which the task belongs (right-justified, zero-filled). Specify an integer value from 1 through 2047.
<code>^USER=^,un</code>	User number (ASCII, left-justified, blank-filled).

Figure 8-24. Q5GETTN Call Format (Sheet 2 of 2)

Q5GETUID - GET USER NUMBER

The Q5GETUID subroutine (refer to figure 8-25) gets the user number under which the job is executing, the amount of execution time available for completion of the job, the account identifier, project number, and privileged status.

The ACCOUNT parameter determines whether the account identifier specified is valid for you, and the MU parameter indicates whether the account identifier specified on the ACCOUNT parameter is a valid master user account by returning the status. If the MU parameter is specified, the ACCOUNT parameter is required.

Q5GETUID uses the Miscellaneous system message.

<u>Call Format</u>	
CALL Q5GETUID({ ^ACCTIME=`,sec } { ^USER=`,user } , optional parameters)
<u>Calling Parameters</u>	
^ACCOUNT=`,acctno	One- to eight-character account identifier to be validated.
<u>Return Parameters</u>	
^ACCTIME=`,sec	Number of system microseconds available to you. ACCTIME= is required if USER= is omitted.
^ERRLEN=`,len	Error message length in bytes (integer).
^MU=`,mu	Return values: Y You are the master user of the account. N You are not the master user of the account.
^PRIV=`,priv	Return values: Y You are a privileged user. N You are a nonprivileged user.
^CACC=`,acctno	acctno is a one- to eight-ASCII-character account identifier under which task is currently executing.
^PROJECT=`,projno	projno is a 1- to 20-alphanumeric-character project number (three-word array) currently in process. This includes the characters * and -.
^ERRMSG=`,msg	80-byte array to which SIL returns an error message.

Figure 8-25. Q5GETUID Call Format (Sheet 1 of 2)

Return Parameters

^FLMAX=^,len	Maximum file length for which you have been validated. len is an integer number of 512-word blocks, ranging from 4 to 1,953,125.
^PRODTN=^,prodt	Return values: Y You are a production user. N You are not a production user.
^STATUS=^,stat	Status code. Possible values: 0 through 202, 250.
^USER=^,un	User number (ASCII, left-justified, 6 bytes). The variable must be on a word boundary. USER= is required if ACCTIME= is omitted.

Figure 8-25. Q5GETUID Call Format (Sheet 2 of 2)

Q5INIT - INITIALIZE CONTROLLEE

The Q5INIT subroutine (refer to figure 8-26) initializes a controllee.

Q5INIT uses the Initialize Controllee system message.

<u>Call Format</u>	
CALL Q5INIT(`LFN=`,lfn,optional parameters)	
<u>Calling Parameters</u>	
`LFN=`,lfn	Name of the controllee file to be initialized. This is a required parameter.
`TLIMIT=`,tl	Time limit for the controller program in microseconds. If TLIMIT is greater than the time limit for the controller, the time limit for the controller is used and no error message is returned. If TLIMIT is omitted, SIL uses the time limit of the calling program.
`WAIT`	Indicates that after the controllee is initialized, the calling task is suspended and the controllee starts executing. If WAIT is omitted, the calling task continues execution after controllee initialization.
<u>Return Parameters</u>	
`DB=`,db	Descriptor block number identifying the initialized controllee. Its descriptor block number can change if the controllee is disconnected.
`ERRLEN=`,len	Error message length in bytes (integer).
`ERRMSG=`,msg	80-byte array in which SIL returns an error message.
`STATUS=`,stat	Status code. Possible values: 0 through 202, 250, 258, 261, 303, 350 through 367, 368, 369, 385, 516, 1726, 1730, 1735.

Figure 8-26. Q5INIT Call Format

Q5INITCH - INITIALIZE CONTROLLEE CHAIN

The Q5INITCH subroutine (refer to figure 8-27) initializes a chain of controllees. The calling task is the controller of the chain.

The maximum number of tasks in a controllee chain is nine. The calling task is already within the controllee chain. A task started by either a batch execute line or an interactive execute line is at level 2 of a controllee chain.

Call Format

CALL Q5INITCH(ˆLFN=ˆ,lfnlist,ˆNTASKS=ˆ,n,optional parameters)

NOTE

The NTASKS= parameter specifies the number of controllee tasks. All other parameters must specify arrays with one word for each controllee task.

Calling Parameters

ˆLFN=ˆ,lfn Array of one through seven ASCII filenames. The names are those of the controllee or drop files that comprise the controllee chain to be initiated. The order of the names in the array is the order of the tasks in the chain. This is a required parameter.

ˆNTASKS=ˆ,n Number of controllee tasks (1 through 7) to be initiated (number of entries in the LFN= array). This is a required parameter.

ˆTMLIMIT=ˆ,t1 Array of integers corresponding to the files in the LFN=array. Each integer indicates the time limit in microseconds for that task. If TMLIMIT is greater than the time limit for its controller, the time limit for its controller is used. If TMLIMIT is omitted, each task is given the time limit of its controller.

Return Parameters

ˆCEDB=ˆ,db Array of integers corresponding to the files specified in the LFN=array. Each integer returned is the descriptor block number of the corresponding task's controllee.

ˆCRDB=ˆ,db Array of integers corresponding to the files specified in the LFN=array. Each integer returned is the descriptor block number of the corresponding task's controller.

ˆDB=ˆ,db Array of integers corresponding to the files specified in the LFN=array. Each integer returned is the descriptor block number of the corresponding task.

Figure 8-27. Q5INITCH Call Format (Sheet 1 of 2)

Return Parameters

<code>^ERRLEN=^,len</code>	Array of integers corresponding to the files in the LFN= array. Each integer returned is the error message length for the corresponding message in the ERRMSG= array.
<code>^ERRMSG=^,msg</code>	Array of 80-byte arrays corresponding to the files specified in the LFN= array. The error message corresponding to the file is returned in the appropriate 80-byte array.
<code>^LEVEL=^,lev</code>	Array of integers corresponding to the files specified in the LFN= array. Each integer returned is the absolute level of the corresponding controllee task.
<code>^STATUS=^,stat</code>	Array of status codes. Possible values: 0 through 202, 250, 258, 303, 350 through 361, 367, 368, 369, 384, 385, 516, 520, 1726, 1730, 1735.

Figure 8-27. Q5INITCH Call Format (Sheet 2 of 2)

Q5LFIHIR - LIST FILE INDEX ENTRY BY HIERARCHICAL SEARCH

The Q5LFIHIR subroutine (refer to figure 8-28) copies the file index entry of an attached file to a buffer SIL defines. To retrieve information from the copied file index entry, call the Q5DCDPFI subroutine.

Q5LFIHIR determines the entry it copies by the qualifiers specified on the call and by a hierarchical search. Each qualifier specified on the call must match the corresponding field in the file index entry.

The file index is searched in the following steps:

1. Private file entries are searched.
2. If the entry is not for a private file, pool file entries are searched in the order in which the task attached the pools. The system pool, if attached, is searched last.
3. If the entry is not for a private file or a pool file, public file entries are searched.
4. If the entry is not for a private, pool, or public file, Q5LFIHIR returns an error status.

Q5LFIHIR copies the entry only if the file is attached.

The buffer used by the Q5LFIHIR routine is the same buffer used by the Q5GETPFI, Q5LFIPOL, Q5LFIPRI, and Q5LFIPUB routines. A call to any of these routines overwrites the buffer contents.

Because Fortran I/O uses the Q5LFIXXX routines, invoking Fortran I/O will cause the buffer to be overwritten. All Q5DCDPFI processing following a Q5LFIXXX or Q5GETPFI call must be done without any intervening Fortran I/O calls.

Q5LFIHIR uses the List Unformatted File Index system message.

Call Format

CALL Q5LFIHIR(optional parameters)

Calling Parameters

^DC=^,dc	Disposition code. The copied file index entry must have the specified disposition code. If DC= is omitted, the file disposition code is not a qualifier.
^IN^	Batch input.
^LR^	Print on a 580-12 line printer.
^LS^	Print on a 580-16 line printer.
^LT^	Print on a 580-20 line printer.
^PR^	Print on any available line printer.
^PU^	Punch.
^P1^	Print on a 501 line printer.
^P2^	Print on a 512 line printer.
^SC^	Scratch file; discard at task termination.
^*^	Any disposition code.

Figure 8-28. Q5LFIHIR Call Format (Sheet 1 of 2)

Calling Parameters

<code>^EC=^,ec</code>	External characteristic. The copied file index entry must have the specified external characteristic. If EC= is omitted, the external characteristic is not a qualifier. <code>^26^</code> 026 punch format <code>^29^</code> 029 punch format <code>^80^</code> 80-column binary punch format <code>^B4^</code> BCD 48-character set <code>^B6^</code> BCD 64-character set <code>^A4^</code> ASCII 48-character set <code>^A6^</code> ASCII 64-character set <code>^A9^</code> ASCII 95-character set <code>^*^</code> Any external characteristic
<code>^FNCOUNT=^,n</code>	Number of file names in the LFN= array. If LFN= is specified but FNCOUNT= is not, SIL assumes that LFN= contains only one file name.
<code>^IC=^,ic</code>	File format. The copied file index entry must have the specified internal characteristic. If IC= is omitted, the file format is not a qualifier. <code>^AS^</code> 8-bit ASCII format; ANSI carriage control <code>^BI^</code> Binary format <code>^PA^</code> 8-bit ASCII format; ASCII carriage control <code>^*^</code> Any internal characteristic
<code>^LFN=^,lfn</code>	Array of file names (one to eight ASCII characters, left justified, blank filled). The copied file index entry must have one of the specified file names. If LFN= is omitted, the file name is not a qualifier.
<code>^ST=^,st</code>	Site identifier. The copied file index entry must have the specified site identifier. If ST= is omitted, the site identifier is not a qualifier. The site determines the site identifiers.
<code>^STRING^</code>	Indicates that the entries in the LFN= array are strings of characters. The file name in the file index entry must begin with one of the strings. If STRING is omitted, Q5LFIHIR assumes that the LFN= array contains file names.
<code>^ZIP=^,zip</code>	Zip code. The zip code field of the copied file index entry must match the specified zip code. If ZIP= is omitted, the zip code is not a qualifier.

Return Parameters

<code>^ERRLEN=^,len</code>	Length of the error message in bytes (integer).
<code>^ERRMSG=^,msg</code>	80-byte array to which SIL returns an error message.
<code>^NFILES=^,n</code>	Number of file index entries returned in the SIL-defined buffer.
<code>^STATUS=^,stat</code>	Status code. SIL returns one of the following values: 0 through 202, 250, 261, 303, 304.

Figure 8-28. Q5LFIHIR Call Format (Sheet 2 of 2)

Q5LFIPOL - LIST POOL FILE INDICES

The Q5LFIPOL subroutine (refer to figure 8-29) copies a set of file index entries to a buffer SIL defines. The set of entries is determined by the qualifiers specified on the call. Each qualifier specified must match the corresponding field in the file index entry.

Q5LFIPOL copies only entries for files that belong to the specified attached pool. The PATTACH control statement and Q5PATACH routine attach pools.

To retrieve information from the copied file index entries, call the Q5DCDPFI subroutine.

The buffer used by the Q5LFIPOL routine is the same buffer used by the Q5GETPFI, Q5LFIHIR, Q5LFIPRI, and Q5LFIPUB routines. A call to any of these routines overwrites the buffer contents.

Because Fortran I/O uses the Q5LFIXXX routines, invoking Fortran I/O will cause the buffer to be overwritten. All Q5DCDPFI processing following a Q5LFIXXX or Q5GETPFI call must be done without any intervening Fortran I/O calls.

Q5LFIPOL uses the List Unformatted File Index system message.

Call Format

CALL Q5LFIPOL(^POOL=^,pool,optional parameters)

Calling Parameters

^ACCOUNT=^,acctno	Account identifier (one to eight ASCII characters). File index entries are returned only for files with the specified account identifier. If this parameter is omitted, the file account identifier is not used to determine the set of files.
^DC=^,dc	Disposition code. File index entries are returned only for files with the specified disposition code. If DC= is omitted, the file disposition codes are not used to determine the set of files. The disposition codes are as follows:
^IN^	Batch input.
^LR^	Print on a 580-12 line printer.
^LS^	Print on a 580-16 line printer.
^LT^	Print on a 580-20 line printer.
^PR^	Print on any available line printer.
^PU^	Punch.
^P1^	Print on a 501 line printer.
^P2^	Print on a 512 line printer.
^SC^	Scratch file; discard at task termination.
^*^	Any disposition code.

Figure 8-29. Q5LFIPOL Call Format (Sheet 1 of 3)

Calling Parameters

^EC=`,ec	External characteristic. File index entries are returned only for files with the specified EC. If EC= is omitted, the file external characteristic is not used to determine the set of files.
^26`	026 punch format
^29`	029 punch format
^80`	80-column binary punch format
^B4`	BCD 48-character set
^B6`	BCD 64-character set
^A4`	ASCII 48-character set
^A6`	ASCII 64-character set
^A9`	ASCII 95-character set
^*`	Any external characteristic
^FNCOUNT=`,n	Number of file names in the LFN= array. If LFN= is specified but FNCOUNT= is not, SIL assumes that one file name is specified.
^IC=`,ic	File format. File index entries are returned only for files with the specified internal characteristic. If IC= is omitted, the file format is not used to determine the set of files.
^AS`	8-bit ASCII format; ANSI carriage control
^BI`	Binary format
^PA`	8-bit ASCII format; ASCII carriage control
^*`	Any internal characteristic
^LFN=`,lfn	Array containing names of pool files for which PFI entries are obtained (ASCII, left justified, blank filled). If LFN= is omitted, file names are not used to determine the set of files.
^MPN=`,mpn	Master project number (one to three alphanumeric characters). File index entries are returned for files with the specified master project number. If this parameter is omitted, the file master project number is not used to determine the set of files.
^POOL=`,pool	Name of the attached pool containing the files for which file index entries are to be obtained. POOL= is a required parameter.
^ST=`,st	Site identifier. File index entries are returned only for files with the specified identifier. If ST= is omitted, the site identifier is not used to determine the set of files. The installation determines the site identifiers.
^STRING`	Indicates that the entries in the LFN= arrays are strings. File index entries are returned only for files that have a name beginning with one of the strings. If STRING is omitted, SIL does not perform string matching.
^ZIP=`,zip	Zip code for the site identifier. If ZIP=,zip is omitted, the zip code is not used to determine the set of files.

Figure 8-29. Q5LFIPOL Call Format (Sheet 2 of 3)

Return Parameters

^ERRLEN=^,len	Length of the error message in bytes (integer).
^ERRMSG=^,msg	80-byte array to which SIL returns an error message.
^NFILES=^,n	Number of file index entries returned in the SIL-defined buffer.
^STATUS=^,stat	Status code. SIL returns one of the following values: 0 through 202, 250, 261, 262, 300, 301, 303, 304.

Figure 8-29. Q5LFIPOL Call Format (Sheet 3 of 3)

Q5LFIPRI - LIST PRIVATE FILE INDICES

The Q5LFIPRI subroutine (refer to figure 8-30) copies a set of file index entries to a buffer SIL defines. The set of entries is determined by the qualifiers specified on the call. Each qualifier specified must match the corresponding field in the file index entry.

With the USER=,un parameter omitted, Q5LFIPRI copies entries for the following private files that meet the specified qualifications.

- Local and permanent files belonging to the task's user number
- Permanent files attached to the job or interactive session, but belonging to another user

With the USER=,un parameter specified, Q5LFIPRI copies entries for private permanent files belonging to the specified user number that the caller can access. (The files must also meet the other qualifications specified on the call.)

To retrieve information from the copied file index entries, call the Q5DCDPFI subroutine.

The buffer used by the Q5LFIPRI routine is the same buffer used by the Q5GETPFI, Q5LFIHIR, Q5LFIPOL, and Q5LFIPUB routines. A call to any of these routines overwrites the buffer contents.

Because Fortran I/O uses the Q5LFIXXX routines, invoking Fortran I/O will cause the buffer to be overwritten. All Q5DCDPFI processing following a Q5LFIXXX or Q5GETPFI call must be done without any intervening Fortran I/O calls.

Q5LFIPRI uses the List Unformatted File Index system message.

Call Format

CALL Q5LFIPRI(optional parameters)

Calling Parameters

ACCOUNT=,acctno	Account identifier (one to eight ASCII characters). File index entries are returned only for files with the specified account identifier. If this parameter is omitted, the file account identifier is not used to determine the set of files.
OACS	Flag determining the access permission set returned by the ACS=,acs parameter on a Q5DCDPFI call. If OACS is specified, the owner's access permission set is returned whether or not the file is attached. If OACS is omitted, the access permission set returned depends on whether the file is attached. If the file is attached, the access permission set in effect is returned. If the file is unattached, the owner's access permission set is returned.
ATTACHED	If ATTACHED is specified, Q5LFIPRI copies file index entries for only those files attached to the executing job. If ATTACHED is omitted, the file need not be attached to the job.
BATCH	If BATCH is specified, Q5LFIPRI copies file index entries for only batch input files. If BATCH is omitted, the file need not be a batch input file.

Figure 8-30. Q5LFIPRI Call Format (Sheet 1 of 3)

Calling Parameters

DC=,dc	Disposition code. File index entries are returned only for files with the specified disposition code. If DC= is omitted, the file disposition code is not used to determine the set of files. The disposition codes are as follows: IN Batch input via an access station. LR Print on a 580-12 line printer. LS Print on a 580-16 line printer. LT Print on a 580-20 line printer. PF Store as a permanent file at access station. PR Print on any available line printer. PU Punch. P1 Print on a 501 line printer. P2 Print on a 512 line printer. SC Scratch file; discard at task termination. * Any disposition code.
EC=,ec	External characteristic. File index entries are returned only for files with the specified EC. If EC= is omitted, the file external characteristic is not used to determine the set of files. 26 026 punch format 29 029 punch format 80 80-column binary punch format B4 BCD 48-character set B6 BCD 64-character set A4 ASCII 48-character set A6 ASCII 64-character set A9 ASCII 95-character set * Any external characteristic
FNCOUNT=,n	Number of file names in the LFN= array. If LFN= is specified but FNCOUNT= is not, SIL assumes that one file name is specified.
IC=,ic	File format. File index entries are returned only for files with the specified internal characteristic. If IC= is omitted, the file format is not used to determine the set of files. AS 8-bit ASCII format; ANSI carriage control BI Binary format PA 8-bit ASCII format; ASCII carriage control * Any internal characteristic
LFN=,lfn	Array containing names of private files for which PFI entries are obtained (ASCII, left justified, blank filled). If LFN= is omitted, filenames are not used to determine the set of files.
MPN=,mpn	Master project number (one to three alphanumeric characters). File index entries are returned for files with the specified master project number. If this parameter is omitted, the file master project number is not used to determine the set of files.
QF	Q5LFIPRI lists only the batch input files that have not yet entered the input queue. If QF is omitted, the queue flag is not used to determine the set of files.

Figure 8-30. Q5LFIPRI Call Format (Sheet 2 of 3)

Calling Parameters

`^ST=^,st` Site identifier. File index entries are returned only for files with the specified identifier. If ST= is omitted, the site identifier is not used to determine the set of files.

`^STRING^` Indicates that the entries in the LFN= arrays are strings. File index entries are returned only for files that have a name that begins with one of the strings. If STRING is omitted, SIL does not perform string matching.

`^USER=^,un` User number (ASCII, left justified, blank-filled). If USER=,un is specified, Q5LFIPRI returns information for files the caller can attach that belong to the specified user number. If USER=,un is omitted, Q5LFIPRI returns information for files the caller owns or has attached.

`^ZIP=^,zip` Zip code for the site identifier. If ZIP=,zip is omitted, the zip code is not used to determine the set of files.

Return Parameters

`^ERRLEN=^,len` Length of the error message in bytes (integer).

`^ERRMSG=^,msg` 80-byte array to which SIL returns an error message.

`^NFILES=^,n` Number of file index entries returned in the SIL-defined buffer.

`^STATUS=^,stat` Status code. SIL returns one of the following values: 0 through 202, 261, 262, 300, 301, 303, 304.

Figure 8-30. Q5LFIPRI Call Format (Sheet 3 of 3)

Q5LFIPUB - LIST PUBLIC FILE INDICES

The Q5LFIPUB subroutine (refer to figure 8-31) copies a set of file index entries to a buffer SIL defines. The set of entries is determined by the qualifiers specified on the call. Each qualifier specified must match the corresponding field in the file index entry.

Q5LFIPUB copies only file index entries for public files.

To retrieve information from the copied file index entries, call the Q5DCDPFI subroutine.

The buffer used by the Q5LFIPUB routine is the same buffer used by the Q5GETPFI, Q5LFIHIR, Q5LFIPOL, and Q5LFIPRI routines. A call to any of these routines overwrites the buffer contents.

Because Fortran I/O uses the Q5LFIXXX routines, invoking Fortran I/O will cause the buffer to be overwritten. All Q5DCDPFI processing following a Q5LFIXXX or Q5GETPFI call must be done without any intervening Fortran I/O calls.

Q5LFIPUB uses the List Unformatted File Index system message.

Call Format

CALL Q5LFIPUB(optional parameters)

Calling Parameters

^ACCOUNT=^,acctno Account identifier (one to eight ASCII characters). File index entries are returned only for files with the specified account identifier. If this parameter is omitted, the file account identifier is not used to determine the set of files.

^DC=^,dc Disposition code. File index entries are returned only for files with the specified disposition code. If DC= is omitted, the file disposition code is not used to determine the set of files. The disposition codes are as follows:

^IN^	Batch input.
^LR^	Print on a 580-12 line printer.
^LS^	Print on a 580-16 line printer.
^LT^	Print on a 580-20 line printer.
^PR^	Print on any available line printer.
^PU^	Punch.
^P1^	Print on a 501 line printer.
^P2^	Print on a 512 line printer.
^SC^	Scratch file; discard at task termination.
^*^	Any disposition code.

Figure 8-31. Q5LFIPUB Call Format (Sheet 1 of 2)

Calling Parameters

<code>^EC=^,ec</code>	External characteristic. File index entries are returned only for files with the specified EC. If EC= is omitted, the file external characteristic is not used to determine the set of files.
<code>^26^</code>	026 punch format
<code>^29^</code>	029 punch format
<code>^80^</code>	80-column binary punch format
<code>^B4^</code>	BCD 48-character set
<code>^B6^</code>	BCD 64-character set
<code>^A4^</code>	ASCII 48-character set
<code>^A6^</code>	ASCII 64-character set
<code>^A9^</code>	ASCII 95-character set
<code>^*^</code>	Any external characteristic
<code>^FNCOUNT=^,n</code>	Number of file names in the LFN= array. If LFN= is specified but FNCOUNT= is not, SIL assumes that one file name is specified.
<code>^IC=^,ic</code>	File format. File index entries are returned only for files with the specified internal characteristic. If IC= is omitted, the file format is not used to determine the set of files.
<code>^AS^</code>	8-bit ASCII format; ANSI carriage control
<code>^BI^</code>	Binary format
<code>^PA^</code>	8-bit ASCII format; ASCII carriage control
<code>^*^</code>	Any internal characteristic
<code>^LFN=^,lfn</code>	Array containing names of public files for which PFI entries are to be obtained (ASCII, left justified, blank filled). If LFN= is omitted, file names are not used to determine the set of files.
<code>^MPN=^,mpn</code>	Master project number (one to three alphanumeric characters). File index entries are returned for files with the specified master project number. If this parameter is omitted, the file master project number is not used to determine the set of files.
<code>^ST=^,st</code>	Site identifier. File index entries are returned only for files with the specified identifier. If ST= is omitted, the site identifier is not used to determine the set of files.
<code>^STRING^</code>	Indicates that the entries in the LFN= arrays are strings. File index entries are returned only for files whose names begin with one of the strings. If STRING is omitted, SIL does not perform string matching.
<code>^ZIP=^,zip</code>	Zip code for the site identifier. If ZIP=,zip is omitted, the zip code is not used to determine the set of files.

Return Parameters

<code>^ERRLEN=^,len</code>	Length of the error message in bytes (integer).
<code>^ERRMSG=^,msg</code>	80-byte array to which SIL returns an error message.
<code>^NFILES=^,n</code>	Number of file index entries returned in the SIL-defined buffer.
<code>^STATUS=^,stat</code>	Status code. SIL returns one of the following values: 0 through 202, 261, 262, 300, 303, 304.

Figure 8-31. Q5LFIPUB Call Format (Sheet 2 of 2)

Q5LSTBUT - LIST BANK UPDATE TABLE

The Q5LSTBUT subroutine (refer to figure 8-32) gets a copy of the Bank Update Table.

Q5LSTBUT issues the List System Table system message.

Call Format

CALL Q5LSTBUT(^BUT=^,but,optional parameters)

Calling Parameters

None.

Return Parameters

^BUT=^,but	32-word array in which SIL returns the bank update table. The array must be a word boundary. This parameter is required.
^ERRLEN=^,len	Error message length in bytes (integer format).
^ERRMSG=^,msg	80-byte array to which SIL returns an error message.
^STATUS=^,stat	Status code. Possible values: 0 through 202, 250.

Figure 8-32. Q5LSTBUT Call Format

Q5LSTCH - LIST CONTROLLEE CHAIN

The Q5LSTCH subroutine (refer to figure 8-33) gets information about one or all tasks in the controllee chain. The subroutine can return the levels, descriptor block numbers, file names, drop file names, and time limits of the tasks in the chain. The descriptor block numbers are useful for identifying a task directly in other SIL calls.

The number of words in the arrays specified to receive information must match the number of tasks in the chain (one through nine). The information returned in the arrays is ordered by level number, beginning with the lowest level (that is, the top of the chain).

Q5LSTCH uses the List Controllee Chain system message.

<u>Call Format</u>	
CALL Q5LSTCH({ `CONTROLE` `CONTROLR` `NLVL=` ,n `PROGRAM` }, optional parameters)
NOTE	
The calling parameters CONTROLLE, CONTROLR, NLVL=,n, and PROGRAM are mutually exclusive. One call parameter must be specified, indicating the tasks within the chain for which information is returned.	
<u>Calling Parameters</u>	
`CONTROLE`	Indicates that SIL obtains information only on the task's controllee.
`CONTROLR`	Indicates that SIL obtains information only on the task's controller.
`NLVL=` ,n	Number of tasks for which information is returned (integer from 2 through 9). All return arrays must be this size.
`PROGRAM`	Indicates that SIL obtains information only on the calling task.
<u>Return Parameters</u>	
`BINARY=` ,lfn	One- to nine-word array in which SIL returns the file names (in ASCII) of the controllees in the chain.
`CEDB=` ,db	One- to nine-word array of integers corresponding to the file names in the BINARY= array. Each integer is the descriptor block number of the corresponding task's controllee.
`CRDB=` ,db	One- to nine-word array of integers corresponding to the file names in the BINARY= array. Each integer is the descriptor block number of the corresponding task's controller.

Figure 8-33. Q5LSTCH Call Format (Sheet 1 of 2)

Return Parameters

^DB=^,db	One- to nine-word array of integers corresponding to the file names in the BINARY= array. Each integer is the descriptor block number of the corresponding task. If the task is interactive, #FF is returned.
^DROPFIL=^,lfn	One- to nine-word array of file names (in ASCII) corresponding to the file names returned in the BINARY= array. Each file name is the name of the drop file for the corresponding task.
^ERRLEN=^,len	Error message length in bytes (integer format).
^ERRMSG=^,msg	80-byte array to which SIL returns an error message.
^ISUDB=^,db	Descriptor block number of the calling program.
^ISULVL=^,lev	Level of the calling program.
^LEVEL=^,lev	One- to nine-word array of integers corresponding to the file names returned in the BINARY= array. Each integer is the level of the corresponding task.
^RNLVL=^,n	Number of levels the call returned in the LEVEL= array.
^STATUS=^,stat	Status code. Possible values: 0 through 202, 250.
^TMLIMIT=^,tl	One- to nine-word array of integers corresponding to the file names returned in the BINARY= array. Each integer is the time limit in microseconds for the corresponding task.

Figure 8-33. Q5LSTCH Call Format (Sheet 2 of 2)

Q5LSTSTB - LIST STATISTICS BUFFER

The Q5LSTSTB subroutine (refer to figure 8-34) gets a copy of the statistics buffer. The statistics buffer is a system table that can contain system performance data.

Q5LSTSTB uses the List System Table system message.

<u>Call Format</u>	
CALL Q5LSTSTB(`STB=`,stb,optional parameters)	
<u>Calling Parameters</u>	
None.	
<u>Return Parameters</u>	
`ERRLEN=`,len	Error message length in bytes (integer).
`ERRMSG=`,msg	80-byte array to which SIL returns an error message.
`STATUS=`,stat	Status code. Possible values: 0 through 202, 250.
`STB=`,stb	100-word array in which SIL returns the statistics buffer. The array must be on a word boundary. This is a required parameter.

Figure 8-34. Q5LSTSTB Call Format

Q5LSTTCB - LIST TIMECARD BUFFER

The Q5LSTTCB subroutine (refer to figure 8-35) gets a copy of the timecard buffer. The timecard buffer is a buffer that can contain accounting information.

Q5LSTTCB uses the List System Table system message.

Call Format

CALL Q5LSTTCB(^TCB=^, tcb, optional parameters)

Calling Parameters

None.

Return Parameters

^ERRLEN=^, len	Error message length in bytes (integer format).
^ERRMSG=^, msg	80-byte array to which SIL returns an error message.
^STATUS=^, stat	Status code. Possible values: 0 through 299.
^TCB=^, tcb	512-word array in which SIL returns the timecard buffer. The array must be on a word boundary. This is a required parameter.

Figure 8-35. Q5LSTTCB Call Format

Q5MEMORY - ALLOCATE STATIC STACK

A Q5MEMORY call (refer to figure 8-36) allocates memory space for a static stack.

A stack is a data structure in which data is added from the top down. Q5MEMORY allocates free space from the high address #800000000000 to a lower address.

The Q5MEMORY call specifies the number of words to be allocated. Q5MEMORY returns the starting address of the stack. For example, if the Q5MEMORY call requests a 512-word stack, the starting address returned is #7FFFFFFF8000.

Q5MEMORY maps in drop file space for the stack, using as few drop file map entries as possible. After the space is allocated, it cannot be freed.

Q5MEMORY calls both the Q5DCDMSC and Q5MAPIN routines.

If 'NOMAP' is specified, Q5MEMORY allocates space starting at #700000000000 to a high address, with no mapping of the drop file.

<u>Call Format</u>	
CALL Q5MEMORY('WRDS=',wrds,'ADDR=',addr,optional parameters)	
<u>Calling Parameters</u>	
'WRDS=',wrds	Number of words to allocate in the stack. This is a required parameter.
'NOMAP'	Space that is allocated for which no mapping of the drop file is done.
<u>Return Parameters</u>	
'ADDR=',addr	Hexadecimal bit address of the start of the stack (integer). The address is always on a double-word boundary. This is a required parameter. If 'NOMAP' is specified, addr is always on a small page boundary.
'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	80-byte array in which SIL returns an error message.
'STATUS=',stat	Status code (integer). Possible values: 0 through 202, 250, 505, 1516, 1519, 1537, 1539, 1550, 1553, 1559.

Figure 8-36. Q5MEMORY Call Format

Q5RECALL - SUSPEND TASK EXECUTION

The Q5RECALL subroutine (refer to figure 8-37) suspends task execution for a minimum specified length of time.

Q5RECALL uses the Recall system message.

Call Format

CALL Q5RECALL(optional parameters)

Calling Parameters

^TIME=^,sec Minimum number of seconds (0 through 2100) that SIL suspends task execution. If TIME= is omitted, SIL suspends the task for a minimum of 30 seconds.

Return Parameters

^ERRLEN=^,len Error message length in bytes (integer).
^ERRMSG=^,msg 80-byte array to which SIL returns an error message.
^STATUS=^,stat Status code. Possible values: 0 through 202, 250, 410.

Figure 8-37. Q5RECALL Call Format

Q5REPREV - ENABLE OR DISABLE USER REPRIEVE

The Q5REPREV subroutine (refer to figure 8-38) either enables or disables user reprieve processing.

User reprieve processing is performed when a task terminates. If enabled, the reprieve subroutine is called whether task termination is normal or abnormal.

Only one reprieve subroutine can be enabled for a task.

To enable user reprieve processing, the Q5REPREV call must specify an entry point within the reprieve subroutine. The entry point is called when the task terminates. The subroutine must be declared external.

NOTE

The reprieve subroutine must return control to the system with a Q5TERM call.

<u>Call Format to Enable User Reprieve</u>	
CALL Q5REPREV (^SUBNAME=^,sub,optional parameters)	
<u>Call Format to Disable User Reprieve</u>	
CALL Q5REPREV(optional parameters)	
<u>Calling Parameter</u>	
^SUBNAME=^,sub	Indicates that user reprieve processing is to be enabled and names the entry point called for user reprieve processing. The name must be declared external within the task calling Q5REPREV. This parameter is required when enabling user reprieve processing; it must not be specified when disabling user reprieve processing.
<u>Return Parameters</u>	
^ERRLEN=^,len	Error message length in bytes (integer).
^ERRMSG=^,msg	80-byte array to which SIL returns an error message.
^STATUS=^,stat	Status code. Possible values: 0, 201, 378, 381, 382.

Figure 8-38. Q5REPREV Call Format

Q5REPREV gets the entry point address, the data base address, and the data base length for the reprieve subroutine and saves the information for use during termination processing. During termination processing, the information is passed to the system by a User Reprieve system message.

The user reprieve routine can perform cleanup processing for the task. If the task aborts because of a time limit error, the user reprieve routine is given an additional one-half of a system second for cleanup processing.

User reprieve processing is performed after any ATC processing that may be performed (refer to Abnormal Termination Processing in chapter 3 of this manual).

If a site accounting routine enables the user reprieve option, user reprieve processing is not available to you.

Q5RFI - RETURN FROM INTERRUPT SUBROUTINE

The Q5RFI subroutine (refer to figure 8-39) returns control from an interrupt subroutine to the interrupted task. Choose one of the following processing options for the interrupted task.

- Abort at the point of the original interrupt.
- Continue processing at the point of the original interrupt.
- Continue processing at a specified entry point.

If you choose to abort processing or if the task aborts after control is returned from the interrupt subroutine, you receive the normal dump and traceback information; however, the interrupt subroutine is not shown in the traceback information. If you specify the RFISUB= parameter on the Q5RFI call and the task subsequently aborts, the traceback information shows the RFISUB= entry point as being called from the original point of interrupt.

Q5RFI issues the Return From Interrupt system message.

<u>Call Format</u>	
CALL Q5RFI(optional parameters)	
<u>Calling Parameters</u>	
^ABORT^	Indicates that the program should abort at the point of the original interrupt. Do not specify both the ABORT and the RFISUB= parameters. If neither is specified, the program continues processing at the point of interruption.
^RFISUB=^,sub	Entry point name (in ASCII) at which processing continues. The entry point must be declared external in the interrupted program. When a fatal error occurs, the point of interrupt appears to call the entry point. Do not specify both the ABORT and RFISUB= parameters. If neither is specified, the program continues processing at the point of interruption.
<u>Return Parameters</u>	
^ERRLEN=^,len	Error message length in bytes (integer).
^ERRMSG=^,msg	80-byte array to which SIL returns an error message.
^STATUS=^,stat	Status code. Possible values: 0, 381, 420.

Figure 8-39. Q5RFI Call Format

Q5RUNBIF - RERUN BATCH INPUT FILE

The Q5RUNBIF subroutine (refer to figure 8-40) informs the system that the specified batch input file is to be rerun if the system fails.

Q5RUNBIF uses the Miscellaneous system message.

The following example of a FORTRAN source line requests that the system rerun the batch input file if the system fails. The name of the batch input file is obtained from a copy of its file index entry via calls to Q5LFIPRI and Q5DCDPFI.

```
CHARACTER*8 LFN  
CALL Q5LFIPRI('BATCH', 'ATTACHED')  
CALL Q5DCDPFI('LFN=', LFN)  
CALL Q5RUNBIF('LFN=', LFN)
```

Call Format

```
CALL Q5RUNBIF('LFN=', lfn, optional parameters)
```

Calling Parameters

'LFN=', lfn Name of the batch input file to be rerun. The name must be left justified and blank filled in a full word on a word boundary. This is a required parameter. You can determine the name of the batch input file by specifying the LFN= parameter on a Q5DCDPFI call.

Return Parameters

'ERRLEN=', len Error message length in bytes (integer).
'ERRMSG=', msg 80-byte array to which SIL returns an error message.
'STATUS=', stat Status code. Possible values: 0 through 262, 303, 400.

Figure 8-40. Q5RUNBIF Call Format

Q5SETLP - CHANGE CURRENT LARGE PAGE LIMIT

The Q5SETLP subroutine (refer to figure 8-41) can change the current large page limit for the task. The specified current large page limit must not exceed the maximum large page limit for the job or task. Determine the maximum large page limit and the current large page limit with a Q5GETLP call.

If the task has more large pages allocated than the specified current large page limit, the contents of the excess large pages are immediately paged out of memory.

Q5SETLP uses the Process System Parameter system message.

The following example of a call sets the current large page limit at six pages.

```
CALL Q5SETLP (^NLP=,6)
```

Call Format

```
CALL Q5SETLP(^NLP=, nlp, optional parameters)
```

Calling Parameter

`^NLP=,nlp` Current large page limit for task (decimal integer).

Return Parameters

`^ERRLEN=,len` Error message length in bytes (integer).

`^ERRMSG=,msg` 80-byte array to which SIL returns an error message.

`^STATUS=,stat` Status code. Possible values: 0 through 202, 250.

Figure 8-41. Q5SETLP Call Format

Q5SNDMCE - SEND MESSAGE TO CONTROLLEE

The Q5SNDMCE subroutine (refer to figure 8-42) sends a message to the calling task's controllee.

If the controllee is a compiled FORTRAN program that has been initialized but whose execution has not begun, the first message sent to the controllee must be for reassignment of the files named in the PROGRAM statement (refer to Execution-Time File Reassignment in the CYBER 200 FORTRAN Language 2 Reference Manual).

Q5SNDMCE uses the Send Message to Controllee system message.

Call Format

CALL Q5SNDMCE('MSG=',msg,optional parameters)

Calling Parameters

'DB=',db	Descriptor block number identifying the controllee to receive the message. If DB= is omitted, the message goes to the next lower controllee in the chain.
'EOF'	Indicates that the ASCII end-of-file character (#1C) is to be appended to the end of the message. EOR, EOG, and EOF are mutually exclusive.
'EOG'	Indicates that the ASCII end-of-group character (#1D) is to be appended to the end of the message. EOR, EOG, and EOF are mutually exclusive.
'EOR'	Indicates that the ASCII end-of-record character (released value, #1F) is to be appended to the end of the message. EOR, EOG, and EOF are mutually exclusive.
'LEN=',len	Length in bytes of the message. If LEN= is omitted, SIL assumes that the first character of the message is a delimiter and that the message consists of the second character through the character preceding the next occurrence of the delimiter. The length does not include any specified delimiting character (EOR, EOG, or EOF) unless the length equals or exceeds 2000. In that case, the delimiting character replaces byte 2000 of the message. SIL sends a maximum of 2000 bytes. If the message exceeds that length, SIL truncates it but does not return an error.
'MSG=',msg	Message to be sent. This parameter is required.
'REJECT'	Indicates that SIL should return an error code if the message cannot be sent immediately. If REJECT is omitted, this message replaces any existing message.

Figure 8-42. Q5SNDMCE Call Format (Sheet 1 of 2)

Return Parameters

^ERRLEN=^,len	Error message length in bytes (integer).
^ERRMSG=^,msg	80-byte array in which SIL returns an error message.
^STATUS=^,stat	Status code. Possible values: 0 through 202, 250, 261, 320, 321, 325.

Figure 8-42. Q5SNDMCE Call Format (Sheet 2 of 2)

Q5SNDMCR - SEND MESSAGE TO CONTROLLER

The Q5SNDMCR subroutine (refer to figure 8-43) sends a message to the calling task's controller. If the controller is the batch processor, the message is written in the task's job dayfile. If the task was initiated at an interactive terminal, the message is sent to the terminal.

Q5SNDMCR uses the Send Message to System Controller system message.

Call Format

CALL Q5SNDMCR(‘MSG=’,msg,optional parameters)

Calling Parameters

- ‘DB=’,db Descriptor block number identifying the controller to receive the message. If DB= is omitted, the message goes to the next higher controller in the chain.
- ‘EOF’ Indicates that the ASCII end-of-file character (#1C) is to be appended to the end of the message. EOR, EOG, and EOF are mutually exclusive.
- ‘EOG’ Indicates that the ASCII end-of-group character (#1D) is to be appended to the end of the message. EOR, EOG, and EOF are mutually exclusive.
- ‘EOR’ Indicates that the ASCII end-of-record character (released value, #1F) is to be appended to the end of the message. EOR, EOG, and EOF are mutually exclusive.

Calling Parameters

- ‘LEN=’,len Length in bytes of the message. If LEN= is omitted, SIL assumes that the first character of the message is a delimiter and that the message consists of the second character through the character preceding the next occurrence of the delimiter.
- The length does not include any specified delimiting character (EOR, EOG, or EOF) unless the length equals or exceeds 2000. In that case, the delimiting character replaces byte 2000 of the message.
- SIL sends a maximum of 2000 bytes. If the message exceeds that length, SIL truncates it but does not return an error.
- ‘MSG=’,msg Message (1 through 2000 bytes). This is a required parameter.
- ‘REJECT’ Indicates that if the message replaces an existing message, SIL suspends task execution until the message can be sent. If REJECT is omitted, this message replaces any existing message. If REJECT is specified, RETURN must be omitted.
- ‘RETURN’ Indicates that SIL returns an error code if the message cannot be sent immediately. If RETURN is omitted, this message replaces any existing message. If RETURN is specified, REJECT must be omitted.

Figure 8-43. Q5SNDMCR Call Format (Sheet 1 of 2)

Return Parameters

<code>^ERRLEN=^,len</code>	Error message length in bytes (integer).
<code>^ERRMSG=^,msg</code>	80-byte array in which SIL returns an error message.
<code>^STATUS=^,stat</code>	Status code. Possible values: 0 through 202, 250, 261, 320 through 324.

Figure 8-43. Q5SNDMCR Call (Sheet 2 of 2)

Q5SNDMDF - SEND MESSAGE TO DAYFILE

The Q5SNDMDF subroutine (refer to figure 8-44) can write a message on the job's dayfile, the system dayfile, or both.

All users can request the following from a batch job.

- A message written to the job's dayfile only
- A message written to both the job's dayfile and the system dayfile

Tasks belonging to privileged user numbers, the installation management user number, the customer engineer user number, and operating system tasks themselves have additional capabilities. These tasks can write on the system dayfile only and can specify the type of entry when writing on the system dayfile only, using the SDFDIAG, SDFLABL, SDFSYSY, and SDFUSER parameters. The system dayfile entry types are described in volume 2 of this manual.

If the specified message is longer than 2000 bytes, SIL truncates the message without returning an error code.

Illegal characters (#00 through #1E, #7F through #FF) are changed to blanks, except for #0DOA, which is changed to #201F. The system adds an end-of-line character (#1F) if none is specified.

If this message fills the job's dayfile, the system replaces it with the message DAYFILE FULL. After that is written, no more messages can be written on the job's dayfile.

When the system dayfile is full, it is renamed Ddddnn (ddd is the day the file became the system dayfile and nn is a sequence number). The backup system dayfile is renamed and becomes the current system dayfile. A new backup system dayfile is created. (The names of the current and backup system dayfiles are specified during system installation.)

Q5SNDMDF uses the Send Message to Dayfile system message.

The following Q5SNDMDF call writes the message THIS IS A MESSAGE on the job's dayfile.

```
CALL Q5SNDMDF('MSG=', '*THIS IS A MESSAGE*')
```

Call Format

CALL Q5SNMDF(‘MSG=’,msg,optional parameters)

NOTE

The BOTH, SDFDIAG, SDFLABL, SDFSYST, and SDFUSER parameters are mutually exclusive. If none of these parameters is specified, the message is written on the job’s dayfile only.

Calling Parameters

‘LEN=’,len Message length in bytes. If LEN= is omitted, SIL assumes that the first character of the message is a delimiter and that the message consists of the second character through the character preceding the next occurrence of the delimiter. SIL sends a maximum of 2000 bytes after removing the delimiters. If the message exceeds that length, SIL truncates it but does not return an error status.

‘MSG=’,msg Message to be sent. This parameter is required.

‘BOTH’ Indicates that the message is written on both the job’s dayfile and the system dayfile. It is written as a USER type entry.

Calling Parameters for Tasks Having Additional Capabilities

‘SDFDIAG’ Indicates that the message is written on the system dayfile only and that it is written as a diagnostic (DIAG) entry. The message must be formatted as a diagnostic entry (excluding time and entry type) as described in volume 2 of this manual. The system generates continuation lines as necessary.

‘SDFLABL’ Indicates that the message is written on the system dayfile only and that it is written as a label (LABL) entry. The message must be formatted as a label entry (excluding time and entry type) as described in volume 2 of this manual. The system generates continuation lines as necessary.

‘SDFSYST’ Indicates that the message is written on the system dayfile only and that it is written as a system (SYST) entry. The message must be formatted as a system entry (excluding time and entry type) as described in volume 2 of this manual. The system generates continuation lines as necessary.

‘SDFUSER’ Indicates that the message is written on the system dayfile only and that it is written as a user (USER) entry. The message must be formatted as a user entry (excluding time and entry type) as described in volume 2 of this manual. The system generates continuation lines as necessary.

Figure 8-44. Q5SNMDF Call Format (Sheet 1 of 2)

Return Parameters

^ERRLEN=^,len	Error message length in bytes (integer).
^ERRMSG=^,msg	80-byte array to which SIL returns an error message.
^STATUS=^,stat	Status code. Possible values: 0 through 202, 250, 261, 310, 320, 327 through 330.

Figure 8-44. Q5SNDMDF Call Format (Sheet 2 of 2)

Q5SNDMJC - SEND MESSAGE TO JOB CONTROLLER

The Q5SNDMJC subroutine (refer to figure 8-45) sends a message to the calling task's job controller (batch processor or virtual system interactive processor). If the job controller is the batch processor, the message is written in the job dayfile. If the task was initiated at an interactive terminal, the message is sent to the terminal.

Q5SNDMJC uses the Send Message to Controller system message.

<u>Call Format</u>	
CALL Q5SNDMJC(‘MSG=’,msg,optional parameters)	
<u>Calling Parameters</u>	
‘EOF’	Indicates that the ASCII end-of-file character (#1C) is to be appended to the end of the message. EOR, EOG, and EOF are mutually exclusive.
‘EOG’	Indicates that the ASCII end-of-group character (#1D) is to be appended to the end of the message. EOR, EOG, and EOF are mutually exclusive.
‘EOR’	Indicates that the ASCII end-of-record character (released value, #1F) is to be appended to the end of the message. EOR, EOG, and EOF are mutually exclusive.
‘LEN=’,len	Length in bytes of the message. If LEN= is omitted, SIL assumes that the first character of the message is a delimiter and that the message consists of the second character through the character preceding the next occurrence of the delimiter. The length does not include any specified delimiting character (EOR, EOG, or EOF) unless the length equals or exceeds 2000. In that case, the delimiting character replaces byte 2000 of the message. SIL sends a maximum of 2000 bytes. If the message exceeds that length, SIL truncates it but does not return an error.
‘MSG=’,msg	Message to be sent. This parameter is required.
‘REJECT’	Indicates that if the message replaces an existing message, task execution is suspended until the message can be sent. If REJECT is omitted, this message replaces any existing message. If REJECT is specified, RETURN must be omitted.
‘RETURN’	Indicates that if the message replaces an existing message waiting for the controller, SIL returns an error code of 323, 324, 332, or 337 (refer to appendix B of this manual). If RETURN is specified, REJECT must be omitted.

Figure 8-45. Q5SNDMJC Call Format (Sheet 1 of 2)

Return Parameters

^ERRLEN=^,len	Error message length in bytes (integer).
^ERRMSG=^,msg	80-byte array in which SIL returns an error message.
^STATUS=^,stat	Status code. Possible values: 0 through 202, 250, 261, 320, 321, 323, 324, 337.

Figure 8-45. Q5SNDMJC Call Format (Sheet 2 of 2)

Q5SNDMJS - SEND MESSAGE TO JOB SESSION

The Q5SNDMJS subroutine (refer to figure 8-46) allows a privileged task to send a message to a job session (for example, an interactive terminal or a batch job dayfile). If the specified message exceeds 2000 characters, SIL truncates the message and issues a warning message.

Call Format

CALL Q5SNDMJS(ˆJDN=ˆ,jdn,ˆUSER=ˆ,user,ˆJOBNAME=ˆ,jobname,ˆMSG=ˆ,msg,
optional parameters)

Calling Parameters

ˆJDN=ˆ,jdn	Job descriptor number (integer value from 1 through 2047) of the job session to which the message is to be sent. This parameter is required.
ˆUSER=ˆ,user	User number (ASCII, left justified, blank filled) associated with the jdn. This parameter is required.
ˆJOBNAME=ˆ,jobname	Job name (ASCII, left justified, blank filled) associated with the jdn. This parameter is required.
ˆMSG=ˆ,msg	Message (1 through 2000 bytes). This parameter is required.
ˆLEN=ˆ,len	Length (integer) in bytes of the message. If LEN is not specified, SIL assumes that the first character of the message is a delimiter and that the message consists of the second character through the character preceding the next occurrence of the delimiter. If the delimiter is not found by the 2000th byte, the delimiting character replaces byte 2000 of the message. SIL sends a maximum of 2000 bytes. If the message exceeds that length, SIL truncates it and reports an error.

Return Parameters

ˆERRMSG=ˆ,errmsg	80-byte array in which SIL returns an error message.
ˆERRLEN=ˆ,errlen	Error message length in bytes (integer).
ˆSTATUS=ˆ,status	Status code (integer). A status code of zero indicates that the message was sent successfully. Possible error codes: 1 through 202, 250, 261, 310, 324, 509, 511.

Figure 8-46. Q5SNDMJS Call Format

Q5SNDMOP - SEND MESSAGE TO OPERATOR

The Q5SNDMOP subroutine (refer to figure 8-47) sends a message to an operator. SIL adds the appropriate control characters to the message. If the specified message is longer than 54 characters, SIL truncates the message without returning an error code.

The operator can be either the local CYBER 200 MCU operator or a remote operator. The REMOTE parameter specifies that the remote operator receive the message. The remote operator is the user logged in with the remote operator user number. The remote operator user number is intended for use by the operator of a remote system.

Call Format

CALL Q5SNDMOP(ˆMSG=ˆ,msg,optional parameters)

Calling Parameters

ˆEOFˆ	Indicates that the ASCII end-of-file character (#1C) is to be appended to the end of the message. EOR, EOG, and EOF are mutually exclusive.
ˆEOGˆ	Indicates that the ASCII end-of-group character (#1D) is to be appended to the end of the message. EOR, EOG, and EOF are mutually exclusive.
ˆEORˆ	Indicates that the ASCII end-of-record character (released value, #1F) is to be appended to the end of the message. EOR, EOG, and EOF are mutually exclusive.
ˆLEN=ˆ,len	<p>Length in bytes of the message. If LEN= is omitted, SIL assumes that the first character of the message is a delimiter and that the message consists of the second character through the character preceding the next occurrence of the delimiter.</p> <p>The length does not include any specified delimiting character (EOR, EOG, or EOF) unless the length equals or exceeds 80. In that case, the delimiting character replaces byte 80 of the message.</p> <p>SIL sends a maximum of 54 bytes. If the message exceeds that length, SIL truncates it but does not return an error. The operator display can show only 54 characters per line.</p>
ˆMSG=ˆ,msg	Message to be sent. This parameter is required.
ˆREMOTEˆ	Indicates that the message is to be sent to the remote operator. If REMOTE is omitted, the message is sent to the local operator. REMOTE and SAVE are mutually exclusive.
ˆRETURNˆ	Indicates that if the message replaces an existing message, SIL returns a status code of 0326. If RETURN is omitted, SIL suspends task execution until it can send the message.
ˆSAVEˆ	Indicates that SIL saves the message for response by the operator. The message is displayed in the O display. If SAVE is omitted, the message is displayed on the local operator K display. REMOTE and SAVE are mutually exclusive.

Figure 8-47. Q5SNDMOP Call Format (Sheet 1 of 2)

Return Parameters

ERRLEN=,len	Error message length in bytes (integer).
ERRMSG=,msg	80-byte array to which SIL returns an error message.
STATUS=,stat	Status code. Possible values: 0 through 202, 250, 261, 320, 326.

Figure 8-47. Q5SNDMOP Call Format (Sheet 2 of 2)

If you do not specify the SAVE parameter on the Q5SNDMOP call, Q5SNDMOP queues the message to be sent to the operator's K display and then returns control to the task so it can resume execution. However, if the system message buffer is full, Q5SNDMOP cannot queue the message and returns abnormal status (status code 0326) to the caller.

If you specify the SAVE parameter on the Q5SNDMOP call, Q5SNDMOP suspends the task until the operator accesses his O display, sees the message, and enters a CFO command in response to the message. The CFO command causes Q5SNDMOP to return control to the task, which can then get the operator response by calling Q5GETMOP.

If the Q5SNDMOP call specifies the REMOTE parameter and the remote operator user number is logged in, Q5SNDMOP sends the message to the remote operator's terminal. If the remote operator user number is not logged in, Q5SNDMOP returns a fatal error (status code 326).

Q5SNDMOP uses the Send Message to Operator system message.

Q5SNDSTR - START CONTROLLEE EXECUTION

The Q5SNDSTR subroutine (refer to figure 8-48) starts a controllee task. You previously initialized the controllee with a Q5INIT call.

Q5SNDSTR uses the Send Message to Controllee system message, although it does not send a message to the controllee.

Call Format

CALL Q5SNDSTR(optional parameters)

Calling Parameters

^DB=^,db Descriptor block number identifying the controllee to be started.
 If DB= is omitted, SIL starts the next lower controllee in the
 controllee chain.

Return Parameters

^ERRLEN=^,len Error message length in bytes (integer).
^ERRMSG=^,msg 80-byte array to which SIL returns an error message.
^STATUS=^,stat Status code. Possible values: 0 through 202, 250, 321.

Figure 8-48. Q5SNDSTR Call Format

Q5TERM - TERMINATE TASK

The Q5TERM subroutine (refer to figure 8-49) terminates a task and its lower-level controllees.

If you do not specify the RESTART parameter, Q5TERM calls an SIL subroutine to close all task files.

Q5TERM uses the Terminate system message.

<u>Call Format</u>	
CALL Q5TERM(optional parameters)	
<u>Calling Parameters</u>	
^ABORT^	Indicates a termination state of #3D (task aborted). ABORT overrides RESTART if both are specified. If ABORT is omitted, the termination state is #3E (normal termination). Refer to Q5GETCTS call description.
^ERROR^	Indicates that the system return code is 4 (nonfatal errors). ERROR and FATAL are mutually exclusive. If ERROR and FATAL are omitted, the system return code is 0 (no errors).
^FATAL^	Indicates that the system return code is 8 (fatal errors). ERROR and FATAL are mutually exclusive. If ERROR and FATAL are omitted, the system return code is 0 (no errors).
^RESTART^	Indicates that the drop file, scratch files, and output files are to be saved so that the program can be restarted. If RESTART is omitted, the files are not saved. If restarted, the program restarts at the point of termination.
^RESUME=^,adr	Virtual bit address at which the restarted program should resume execution. The address specified must be in the same subroutine that issued the Q5TERM call.
<u>Return Parameters</u>	
^ERRLEN=^,len	Error message length in bytes (integer).
^ERRMSG=^,msg	80-byte array in which SIL returns an error message.
^STATUS=^,stat	Status code. Possible values: 0 through 199, 261.

Figure 8-49. Q5TERM Call Format

Q5TERMCE - DISCONNECT CONTROLLEE

The Q5TERMCE subroutine (refer to figure 8-50) disconnects a previously initialized controllee.

Q5TERMCE uses the Disconnect Controllee system message.

Call Format

CALL Q5TERMCE(optional parameters)

Calling Parameters

None.

Return Parameters

^ERRLEN=,len Error message length in bytes (integer).
^ERRMSG=,msg 80-byte array in which SIL returns an error message.
^STATUS=,stat Status code. Possible values: 0 through 202, 261, 370.

Figure 8-50. Q5TERMCE Call Format

Q5TIME - GET SYSTEM TIME

The Q5TIME subroutine (refer to figure 8-51) gets the system time and date. Specify the TIME=, DATE=, JULIAN=, or MASTER= parameter on the call.

Q5TIME uses the Miscellaneous system message.

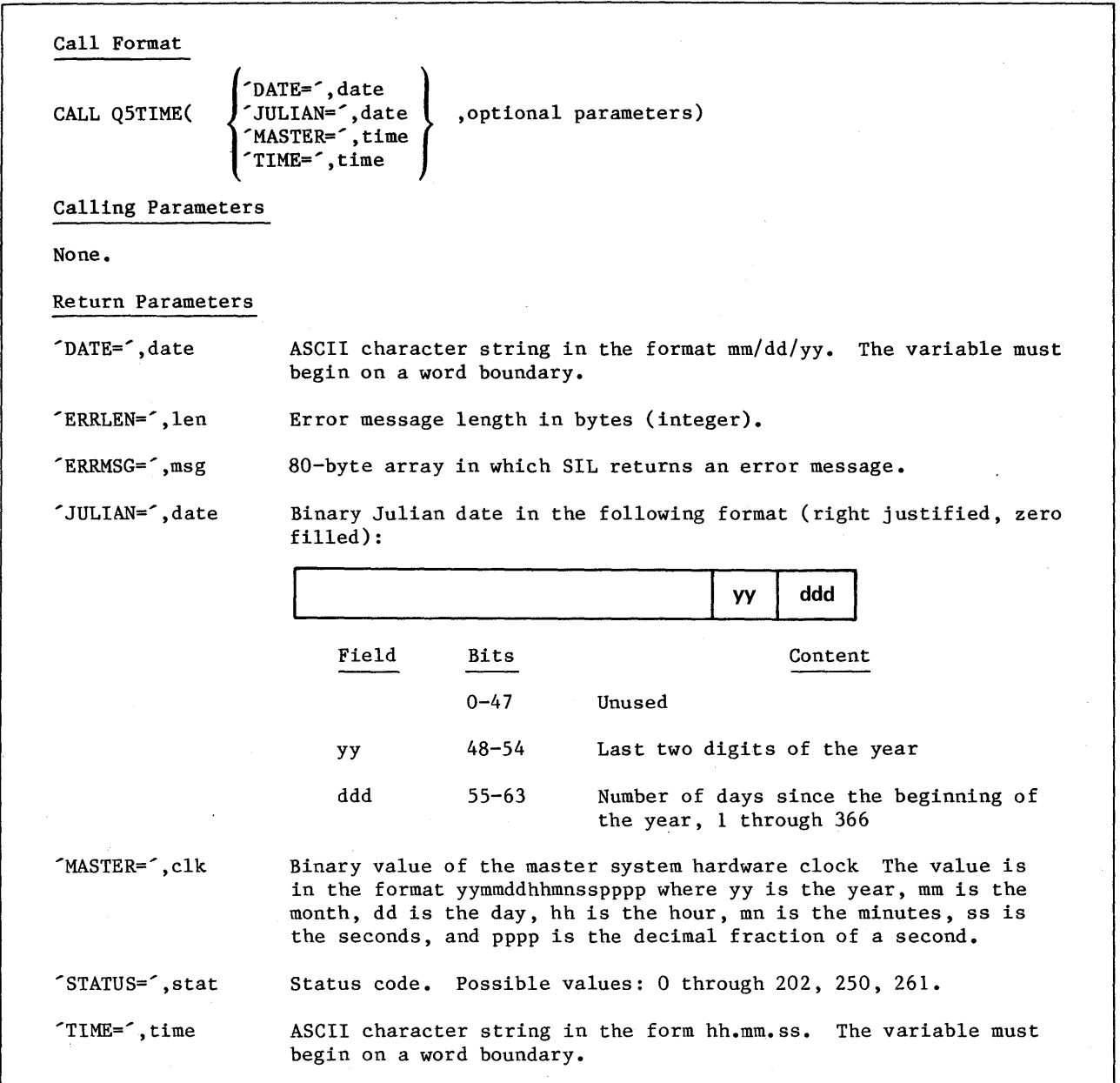


Figure 8-51. Q5TIME Call Format

Q5VRACC - CHANGE ACCOUNTING RATE

The Q5VRACC subroutine (refer to figure 8-52) changes the accounting rate for the task.

If the specified variable rate entry has a password, a Q5VRACC call specifying the entry and its password is valid for any user. If the specified password does not match the password in the entry, Q5VRACC returns a fatal error (status code 0473).

If the specified variable rate entry does not have a password, a Q5VRACC call is valid for a public controllee or a controllee executed for a user number having the variable rate accounting permission.

The site can set an installation parameter that prevents accounting rate changes.

The site can also specify a test routine to restrict use of variable accounting rates. If the call fails the test, it returns a fatal error (status code 0473).

Q5VRACC uses the Variable Rate Accounting system message.

<u>Call Format</u>	
CALL Q5VRACC(optional parameters)	
<u>Calling Parameters</u>	
^AAF^	Site accounting flag. A site accounting interface routine can specify this parameter so that it, and not the task, determines relieve processing. If AAF is omitted, the task can enable its own relieve subroutine (refer to the Q5REPREV subroutine description in this chapter).
^PW=^,pw	Password to the specified variable rate entry (eight-character string). If the variable rate entry has a password, it must be specified.
^VRI=^,vri	Index into the variable rate table (integer from 0 through 4096). If VRI= is omitted, SIL uses index 0.
<u>Return Parameters</u>	
^ERRLEN=^,len	Error message length in bytes (integer).
^ERRMSG=^,msg	80-byte array to which SIL returns an error message.
^STATUS=^,stat	Status code. Possible values: 0 through 202, 250, 470 through 473.

Figure 8-52. Q5VRACC Call Format

This chapter describes the SIL routines that perform I/O operations or functions related to file I/O. Chapter 8 describes SIL routines that are not related to file I/O operations.

Table 9-1 lists the routines described in this chapter, grouped according to a shared function. To use the routines, you must be familiar with the file concepts discussed in chapter 2 of this manual.

Table 9-1. SIL I/O Calls (Sheet 1 of 2)

<u>Permanent file access</u>	
Q5DEFINE	Defines a permanent file
Q5ATTACH	Attaches a permanent file
Q5RETURN	Returns a permanent file or discards a temporary file
Q5CHANGE	Changes file attributes
Q5GIVE	Gives file ownership to another user
Q5PURGE	Purges a permanent file
<u>Local file access</u>	
Q5RQUEST	Creates or accesses a local file
Q5CHANGE	Changes file attributes
Q5RETURN	Returns a local file
Q5LABEL	Creates or accesses a local file in a multiple set
<u>Pool access</u>	
Q5PCREAT	Adds a pool to the pool list
Q5GIVE	Gives a file to a pool
Q5PGRACC	Grants access to a pool
Q5PATACH	Attaches a pool of files
Q5PDTACH	Returns a pool of files
Q5PREACC	Removes access to a pool
Q5PURGE	Purges pool file
Q5PDESTR	Removes a pool from the pool list
Q5POOLS	Lists the pools
Q5USERL	Lists the users granted access to a pool
<u>Public file creation</u>	
Q5GIVE	Adds a file to the public file list
<u>FIT manipulation</u>	
Q5GENFIT	Generates a FIT
Q5SETFIT	Changes FIT fields
Q5GETFIT	Retrieves contents of FIT fields
Q5RETFIT	Returns a FIT

Table 9-1. SIL I/O Calls (Sheet 2 of 2)

I/O preparation

Q5OPEN	Opens a file for I/O
Q5GETFIL	Opens or requests and opens a file
Q5CLOSE	Closes a file for I/O
Q5REELSW	Continues processing with next tape volume

Implicit I/O preparation

Q5MAPIN	Associates a virtual address range with a mass storage file
Q5MAPOUT	Disassociates a virtual address range from a mass storage file

Explicit I/O by physical blocks

Q5READ	Reads data from a file
Q5WRITE	Writes data to a file
Q5CHECK	Checks to see whether I/O operation is complete

Explicit tape I/O by blocks

Q5GETB	Reads buffer record
Q5PUTB	Writes buffer record
Q5CHECKB	Checks to see whether the buffer I/O operation is complete

Explicit I/O by logical partitions

Q5GETN	Reads complete partition
Q5GETP	Reads partial partition
Q5PUTN	Writes complete partition
Q5PUTP	Writes partial partition
Q5ENDPAR	Writes partition delimiter

File positioning

Q5REWIND	Rewinds file
Q5SKIP	Skips file partitions forward or backward

Other functions

Q5CLIOER	Clears tape I/O error
Q5REDUCE	Releases allocated mass storage space that is not in use by the file
Q5ROUTE	Routes a file
Q5PERMIT	Changes access permission set

SIL I/O OVERVIEW

To perform file I/O using SIL calls, the program must first issue calls to prepare the file for I/O and then issue calls to perform the I/O operations. SIL calls are also provided to perform cleanup operations after completion of I/O to a file, although the system performs these operations at task completion if the program does not perform them itself.

PREPARING A FILE FOR I/O

To prepare a file for I/O, the program must call SIL routines to perform the following operations:

- Access or create the file if the file is not attached to the job
- Create a FIT for the file
- Open the file

To create a file means to create a file index entry for the file. A program can create or access files using utilities described in chapter 4 or using SIL calls.

The following SIL calls access existing files.

<u>SIL Call</u>	<u>File Type</u>
Q5ATTACH	Private permanent mass storage file
Q5PATACH	Pool mass storage file

The following SIL calls create files.

<u>SIL Call</u>	<u>File Type</u>
Q5DEFINE	Private permanent mass storage file
Q5RQUEST	Local mass storage file or a file connected to a terminal
Q5GETFIL	Local mass storage file or a file connected to a terminal if the file specified on the call does not exist

FIT Processing

To perform I/O operations on a file, the file must have a file information table (FIT) associated with it. The FIT is the table SIL uses to coordinate I/O processing for the file during the task.

When the task creates a file using a Q5DEFINE, Q5RQUEST, or Q5GETFIL call, the call also generates a FIT for the file. If the task does not create the file, it can generate a FIT for the file with the Q5GENFIT call. If a FIT does not exist for the file when it is opened, the Q5OPEN call generates a FIT for the file.

After a FIT exists for a file, the task can reference the file on SIL calls using either the file name or the number SIL assigned to the FIT for the file. The number is called the file logical unit number (flun). File specification by number, instead of by name, is recommended because file specification by name requires that SIL associate the name with a FIT.

The FIT format is in appendix D of this manual. When SIL generates a FIT, it takes field values from the following sources, in this order:

1. Parameter specifications on the SIL call that generates the FIT
2. File index entry field values
3. Default values

The task can change FIT field values with the Q5SETFIT call and retrieve FIT field values with the Q5GETFIT call.

SIL discards FITs when task processing completes. The task can discard a FIT with a Q5RETFIT call or with the RETFIT parameter on the Q5CLOSE or Q5RETURN call.

Opening a File

To open a file for I/O, the task must specify the file on a Q5OPEN or Q5GETFIL call. SIL opens the file for explicit I/O unless the IMP parameter is specified on the call.

The Q5OPEN call can specify the access modes that the task requires. For explicit I/O, the call should specify read, write, modify, and/or append modes, depending on the operations to be performed. For implicit I/O, the call should specify read mode or read and write modes.

EXPLICIT I/O

After opening a file for explicit I/O, the task must issue an SIL call for each read operation from the file or each write operation to it. SIL can perform explicit I/O by logical partition or by physical block.

Explicit I/O by Logical Partition

A task can perform explicit I/O by logical partitions. A logical partition within an F or U format file is a record or the entire file. A logical partition within an R, L, or W format file is a record, a group, or the entire file. A logical partition for a B format file is the entire file.

Explicit I/O by logical partition uses a working storage area and an I/O buffer. Explicit I/O calls copy data between the working storage area and the I/O buffer.

The working storage area is an array the task defines. It must be large enough for the largest amount of data the task reads or writes with one SIL call. The I/O buffer(s) must be assigned by the task via the Q5OPEN or Q5GETFIL call. A Q5SETFIT call may also be used to define the I/O buffer.

Reading Data By Logical Partition

Q5GETN and Q5GETP are the SIL calls that read data by logical partition. When a task calls Q5GETN or Q5GETP to read data, the system copies the physical block of data from the mass storage file to the I/O buffer. Each Q5GETN or Q5GETP call then copies a logical partition of data from the I/O buffer to the working storage area, where the task can reference the data. The system copies new blocks of data to the I/O buffer as required.

Q5GETN copies data to the working storage area until it encounters a partition delimiter. Q5GETP copies data until the specified working storage area is full or until it encounters a partition delimiter.

Q5GETN and Q5GETP do not copy the partition delimiter of the requested partition. The partition delimiters of lower-level partitions are copied. For example, if Q5GETN copies a group, it copies the record delimiters within the group, but it does not copy the group delimiter.

Writing Data By Logical Partition

Q5PUTN and Q5PUTP are the SIL calls that write data by logical partition. Similar to Q5GETN and Q5GETP, Q5PUTN and Q5PUTP also use a working storage area and an I/O buffer, but the copying of the data is from the working storage area to the I/O buffer. When the I/O buffer is full, the system copies the data to the storage file.

Q5PUTN copies a full partition of data from the working storage area to the I/O buffer and then terminates the partition. Q5PUTP copies the data in the working storage area but does not terminate the partition unless the TERM parameter is specified on the call. The task can also terminate a partition with a Q5ENDPAR call.

Example of Explicit I/O By Record

The following is an example of a FORTRAN program that uses SIL calls to read and write logical records.

The program uses two files, FILE1 and FILE2. FILE1 is a permanent mass storage file containing R format records; the maximum size of a record is 10 words. FILE2 is a local mass storage file that the program creates.

Using SIL calls, the program reads a logical record from FILE1 and checks to see whether the first word of the record contains a value greater than 500. If it does, it writes the record on FILE2.

When the program reaches the end of the FILE1 data, it closes both files. It then checks a count variable to determine whether it wrote any records to FILE2. If it did, it stores FILE2 as a permanent file.

```

PROGRAM RECIO
COMMON/BUF/BUF1(8192),BUF2(8192)
INTEGER FL1, FL2, WSA(10), COUNT, ISTAT,
+ IERMSG(10)
CHARACTER*8 FP

C
C     PREPARATION FOR FILE I/O
C
CALL Q5ATTACH(^LFN=^,^FILE1^)
CALL Q5RQUEST(^LFN=^,^FILE2^)
CALL Q5OPEN(^LFN=^,^FILE1^,^ACS=^,^R^,^0001=^,BUF1,^BUFL1=^,16,
+ ^RFLUN=^,FL1)
CALL Q5OPEN(^LFN=^,^FILE2^,^ACS=^,^W^,^BUF1=^,BUF2,^BUFL1=^,16,
+ ^RFLUN=^,FL2)

C
COUNT=0
C
READ LOGICAL RECORD
C
1 CALL Q5GETN(^FLUN=^,FL1,^WSA=^,WSA,^WSL=^,80,
+ ^STATUS=^,ISTAT,^ERRMSG=^,IERMSG)
C
CHECK FOR END OF FILE
C
IF (ISTAT .EQ. 1416)
+ GO TO 2
C
DETERMINE IF RECORD IS WRITTEN
C
IF (WSA(1) .GE. 500)
+ CALL Q5PUTN(^FLUN=^,FL2,^WSA=^,WSA,
+ ^WSL=^,80)
COUNT = COUNT+1
GO TO 1
C
CLOSE FILES
C
2 CALL Q5CLOSE(^FLUN=^,FL1)
CALL Q5CLOSE(^FLUN=^,FL2)

C
DETERMINE IF FILE2 IS SAVED
C
IF (COUNT .GT. 0)
+ CALL Q5DEFINE(^FLUN=^,FL2)
STOP
END

```


Explicit I/O By Physical Block

A task can perform explicit I/O by physical block, using the Q5READ, Q5WRITE, and Q5CHECK calls.

A block is 512 contiguous words. Its starting word address must be a multiple of 512. A Q5READ call copies one or more blocks of data from a mass storage file to an I/O buffer in the program space. A Q5WRITE call copies one or more blocks of data from an I/O buffer in program space to a mass storage file.

The FIT for a file can define two I/O buffers for the file. The Q5READ and Q5WRITE calls can specify which buffer the call uses. The buffer must be large enough for the data copied.

Unless the WAIT parameter is specified, SIL performs the read or write operation while execution of the calling task continues. When necessary, the task should issue a Q5CHECK call to determine whether the I/O operation has completed. If the WAIT parameter is specified, SIL suspends task execution until the I/O operation completes.

The task specifies the I/O operation on a Q5CHECK call, using its request number. SIL returns the request number of a Q5READ or Q5WRITE call if the RSN parameter is specified on the call.

Appending Data

Appending data to a file means writing data at the end of the file only. If the record format of the file uses a file delimiter, the data must overwrite the file delimiter.

To append data, a task must perform the following steps:

1. Position the file at its end by reading data until the end of file is read or by skipping to the end of the file with a Q5SKIP call that specifies the 'PART=', 'F' parameter.
2. Position the file before the file delimiter with a Q5SKIP call that specifies the 'COUNT=', -1 parameter to skip backwards over the file delimiter.
3. Write data to the file. The data overwrites the existing file delimiter. A new file delimiter is written when the file is closed.

NOTE

If the existing data on a file was written by Q5PUTN or Q5PUTP calls, do not use a Q5WRITE call to append additional data to the file. Data written by Q5PUTN or Q5PUTP calls could end within a block. Using a Q5WRITE call to append data overwrites the last block of the file if it ends with a partial block.

IMPLICIT I/O

After opening a file for implicit I/O, the task must map the file into a virtual address range. Once mapped in, the system implicitly performs the I/O operations needed to read and write data from the file.

The virtual address range must be declared as an array in a common block that is not mapped into the controllee file. To prevent the mapping in of the common block, specify the common block on a GROS parameter on the LOAD statement for the program.

To map in a file, the task issues a Q5MAPIN call that associates the file with the common block array. The call must specify the name of the file, the subscripted name of the first element of the array, and the length of the array. The array specified must begin on a page boundary.

After the array is associated with a file, a task reference to an array element causes the system to copy the appropriate block of data from the file to the array.

Data can also be stored in the array. If the task has write access to the file, the contents of the array are written to the file when the file is mapped out. A file is mapped out when it is disassociated from its virtual address range. A file open for implicit I/O is mapped out when the file is closed or when a Q5MAPOUT call specifies the file.

Example of Implicit I/O

The following job compiles, loads, and executes a program that uses implicit I/O. The program accesses file space, using a 1,000,000-word array assigned to a common block named BUFFER. The LOAD statement specifies the BUFFER common block on a GROS parameter so that it is not mapped into the controllee file.

The program creates and opens file TEST for implicit I/O and maps the BUFFER array into the file. It then sets the contents of the BUFFER array to zero. The contents of the BUFFER array are implicitly stored in file TEST.

```
FTN200.
LOAD(GROS=*BUFFER,CN=IMPLICIT,L=0)
IMPLICIT.
*EOR
PROGRAM CLEAR
IMPLICIT INTEGER (A - Z)
PARAMETER (MAXFILE=1000000)
DATA LFN / 4HTEST /
DIMENSION BUFFER (MAXFILE)
COMMON / BUFFER / BUFFER
CALL Q5GETFIL ('LFN=', LFN, 'IMP', RLEN=*, FILELEN)
CALL Q5MAPIN ('LFN=', LFN,
+ 'VBA=', BUFFER,
+ 'LEN=', FILELEN)
DO 1 I=1, MAXFILE
1 BUFFER (I) = 0
CALL Q5CLOSE ('LFN=', LFN)
STOP
END
```

The Q5MAPIN call maps in the initial file length (RLEN). However, the file is extended up to MAXFILE words long as the DO loop references space beyond the initial file length.

SIL I/O CALLS

This section contains a figure for each SIL routine. The figure contains a call format specifying the required parameters followed by parameter descriptions.

The parameter descriptions are divided between calling parameters and return parameters. A calling parameter specifies a value used by the SIL routine. A return parameter specifies the name of the variable in which SIL returns a value.

Parameter keywords are listed as FORTRAN literals. Options are listed as lowercase variable names. The available mnemonic values for calling parameter options are listed as FORTRAN literals.

Table 9-2 lists the value ranges for calling parameters commonly used by SIL I/O calls. If a parameter value is specified outside of the valid range, SIL returns abnormal status (status codes 1 through 100).

Table 9-2. Calling Parameter Value Ranges

Parameter	Lower Bound (Inclusive)	Upper Bound (Inclusive)
FLUN=	0	109
MNR=	0	$2^{32} - 1$
MXR=	0	$2^{32} - 1$
PC=	0	255
RMK=	0	255
RP=	1	65535
RSN=	1	6
SLEV=	1	8
BUFL1=	1	48
BUFL2=	1	48
ERL=	0	65 535
WSL=	0	$2^{32} - 1$
REC=	1	$2^{32} - 1$
VRI=	1	255
BUFLEN=	1	6144
BYTCNT=	1	524288
COUNT=	$-(2^{48})$	$2^{48} - 1$

Q5ATTACH - ATTACH PERMANENT FILE

Call the Q5ATTACH routine (refer to figure 9-1) to attach any of the following:

- A private permanent file belonging to the caller
- All private permanent files belonging to the caller
- A private permanent file to which you have access, but which you do not own

If the call attaches only one file, specify the set of access modes to be allowed while the file is attached. You must have permission for each access mode specified. Subsequent calls to open the file can specify any of the allowed access modes.

If, while attaching all of your permanent files, Q5ATTACH encounters an error preventing it from attaching a file, it records the error and continues attaching files; therefore, the status code returned is that of the last error encountered.

Q5ATTACH cannot attach a local mass storage file, or a file connected to a terminal. An attempt to do so returns a fatal error (status codes 0302 or 1502).

When you call Q5ATTACH for a file that has some unavailable segments because one or more devices are down, a fatal error status 1619 is returned.

Q5ATTACH issues the ATTACH FILE system message.

Call Format

CALL Q5ATTACH ({ ^LFN=^,lfn } , optional parameters)

Calling Parameters

^LFN=^,lfn Name of an unattached permanent file. LFN=,lfn must be specified if * is omitted.

^*^ Indicates that SIL attaches all unattached, permanent files belonging to you. * must be specified if LFN=,lfn is omitted. If * is specified, the ACS=,acs and USER=,un parameters cannot be specified.

^ACS=^,acs Access modes allowed while the file is attached (specified by a string composed of the following letters).

R	Read access
W	Write access
X	Execute access
A	Append access
M	Modify access

If ACS=,acs is omitted, only read and execute modes are allowed while the file is attached. The ACS=,acs and * parameters are mutually exclusive.

Figure 9-1. Q5ATTACH Call Format (Sheet 1 of 2)

Calling Parameters

^USER=`,un User number that owns the file (ASCII, left justified blank-filled). If USER=,un is omitted, SIL assumes that the file belongs to the caller. The USER=,un and * parameters are mutually exclusive.

^PARTIAL=`,x Specifies whether or not segments can be attached even if some segments of the file are on down devices. x has the value of Y (yes) or N (no). To use Y, the parameter ACS=R must also be used. N is the default value for x.

Return Parameters

^ERRLEN=`,len Error message length in bytes.

^ERRMSG=`,msg Error message. The variable msg must be 80 bytes long.

^STATUS=`,stat Status code. Possible values: 0 through 199, 204, 205, 258, 302, 303, 1501, 1502, 1504, 1510, 1543, 1617, 1618, 1619, 1682, 1726, 1730, 1735.

Figure 9-1. Q5ATTACH Call Format (Sheet 2 of 2)

Q5CHANGE - CHANGE FILE ATTRIBUTES

Call the Q5CHANGE routine (refer to figure 9-2) to change file attributes. Q5CHANGE records the specified changes in the file's file index entry FILEI and in its FIT. The file attribute changes are permanent for a permanent file.

Unless you are privileged, you can change the attributes of your private files only.

If a parameter on a Q5CHANGE call is not valid for the device type of the file, Q5CHANGE returns a warning error. If the parameter is a mass storage parameter, Q5CHANGE returns status code 1454; if it is a tape parameter, Q5CHANGE returns status code 1472.

Q5CHANGE issues the CHANGE FILE ATTRIBUTES system message.

<u>Call Format</u>	
CALL Q5CHANGE({ ^LFN=,lfn ^FLUN=,rflun } , optional parameters)
<u>Calling Parameters</u>	
^LFN=,lfn	File name. LFN=,lfn must be specified if FLUN=,rflun is omitted.
^FLUN=,rflun	File logical unit number SIL assigned to the FIT. FLUN=,rflun must be specified if LFN=,lfn is omitted.
^MNR=,mnr	Minimum record length in bytes. For record types other than F, SIL checks to see that a record is not shorter than this value. SIL does not use this value when writing F format records. If MNR=,mnr is omitted, SIL does not change the minimum record length.
^MXR=,mxr	Maximum record length. For F format records, the maximum record length is the fixed record length. For other record formats, SIL checks to see that the record length does not exceed this value. If MXR=,mxr is omitted, SIL does not change the maximum record length.
<u>Calling Parameters for Mass Storage Files Only</u>	
^ACCOUNT=,acctno	Account identifier (one to eight ASCII characters). If this parameter is omitted, SIL does not change the account identifier.
^CT=,ct	Front-end communication type. If CT=,ct is omitted, SIL does not change the communication type.
^NRHF	No Remote Host Facility
^RHF	Remote Host Facility
^RWF	Remote Workstation Facility
^DFLEN=,df1	Drop file length in 512-word blocks. If DFLEN=,df1 is omitted, SIL does not change the drop file length.

Figure 9-2. Q5CHANGE Call Format (Sheet 1 of 4)

Calling Parameters for Mass Storage Files Only

FC=,fc	File category. If FC=,fc is omitted, SIL does not change the file category.
B	Batch file
U	User file
D	User-created drop file
IC=,ic	File format. If IC=,ic is omitted, SIL does not change the file format.
AS	8-bit ASCII code; ANSI carriage control if the file is a print file
BI	Binary
PA	8-bit ASCII code; ASCII carriage control if the file is a print file
MPN=,mpn	Master project number (one to three alphanumeric characters). If this parameter is omitted, SIL does not change the master project number.
OSTAT=,ostat	Output-file-family status. If OSTAT=,ostat is omitted, SIL does not change the output-file-family status. Only privileged tasks are allowed to use this parameter.
	0 Normal status.
	1 Destination LID disabled.
	2 Destination not responding.
	3 Destination rejecting file.
	4 SIL error occurred during file transfer.
	5 Diverted.
	6 Hardware path to LID not available.
	7 System error occurred during file transfer.
	8 RHF error occurred during file transfer.
	9 RWF error occurred during file transfer.
	10-27 Reserved by CDC.
	28-31 Reserved for installations.
RP=,rp	Retention period in days. If RP=,rp is omitted, SIL does not change the retention period.
SFO=,sfo	File organization. If SFO=D is specified, the file must be a mass storage file and use the F record format. If SFO=,sfo is omitted, SIL does not change the file organization.
D	Direct access
S	Sequential access
TYPE=,typ	File type. If TYPE=,typ is omitted, SIL does not change the file type.
PD	Physical data file (not executable)
VC	Virtual code file (executable)

Figure 9-2. Q5CHANGE Call Format (Sheet 2 of 4)

Calling Parameters for Mass Storage Files Only

`^AU=^,blocks` Allocation units. The number of 512-word blocks to be allocated when the file is extended. The value range of blocks is 1 to 65,535. If blocks is not a multiple of the DAU for the device at which the first allocation occurs, it is rounded to the next higher multiple of the DAU.

`^USER=^,userno` Owner user number (ASCII, left-justified, blank filled). Valid only if RSTDF= is specified. If omitted, the site security administrator user number is assumed to be the owner.

`^RSTDF=^,df` Specify `df=y` to enable the restart of a drop file that has been flagged by the system as nonrestartable.

The USER= and RSTDF= parameters can only be used by the site security administrator user number (refer to chapter 7 of the Installation Handbook for details).

Calling Parameters for Mass Storage and Tape Files Only

`^BT=^,bt` Blocking type. If `BT=,bt` is omitted, SIL does not change the blocking type.

- `^C^` Character count
- `^I^` Internal (tapes only)
- `^K^` Record count (tapes only)

`^NFNAME=^,nf` New file name. If `NFNAME=,nf` is omitted, SIL does not change the file name.

`^PC=^,x` Padding character used to fill the working storage area. If `PC=,x` is omitted, SIL does not change the padding character.

`^RT=^,rt` Record format. If `SFO=D` is specified, the only valid record format is F. If `RT=,rt` is omitted, SIL does not change the record format.

- `^B^` System block (tapes only)
- `^F^` ANSI fixed length
- `^L^` CYBER Record Manager control word (tapes only)
- `^R^` Record mark delimited
- `^U^` Undefined structure
- `^W^` Control word

`^RMK=^,x` Record mark character. If `RMK=,x` is omitted, SIL does not change the record mark character.

Figure 9-2. Q5CHANGE Call Format (Sheet 3 of 4)

Calling Parameters for Tape Files Only

`^CONVERT=^,tm` Data conversion mode; indicates whether the tape data is to be stored as binary or character data.

`^Y^` Character data; convert data to and from character codes.

`^N^` Binary data; do not convert data.

`^LPROC=^,lp` Label processing option.

`^R^` Read and verify existing labels.

`^W^` Write new labels.

`^MPRU=^,mpru` MPRU size in bytes; applicable to V format files only. If `MPRU=,mpru` is omitted, SIL does not change the MPRU size.

`^RPB=^,rpb` Records per block; applicable to K format files only. If `RPB=,rpb` is omitted, SIL does not change the records per block.

Return Parameters

`^ERRLEN=^,len` Error message length in bytes (integer).

`^ERRMSG=^,msg` Error message. The variable `msg` must be 80 bytes long.

`^STATUS=^,stat` Status code. Possible values: 0 through 199, 204, 303, 308, 0510, 0517, 0519, 1402, 1425, 1429, 1454, 1470 through 1472, 1505, 1586, 1704, 1800, 1802, 1803.

Figure 9-2. Q5CHANGE Call Format (Sheet 4 of 4)

Q5CHECK - CHECK I/O REQUEST STATUS

Call the Q5CHECK routine (refer to figure 9-3) to determine the status of a Q5READ or Q5WRITE request. If more than one Q5READ or Q5WRITE request is outstanding, the request can be identified by its request ordinal. The RSN parameter on the Q5READ or Q5WRITE call returns the request ordinal.

After determining the request status by examining the appropriate fields in the FIT, SIL returns control immediately to you by default. If the WAIT parameter is specified on the Q5CHECK call, however, SIL suspends program execution until the I/O request has completed.

Call Format

```
CALL Q5CHECK( { ^LFN=,lfn  
               ^FLUN=,flun } , ^IOSTAT=,sts, optional parameters)
```

Calling Parameters

^LFN=,lfn File name. LFN=,lfn must be specified if FLUN=,flun is omitted.

^FLUN=,flun Logical number SIL assigned to the file. FLUN=,flun must be specified if LFN=,lfn is omitted.

^RSN=,rsn Request number of the request to be checked. SIL assigns request numbers to identify concurrent I/O request. Q5READ and Q5WRITE return an RSN value. If RSN=,rsn is omitted, SIL checks the last issued I/O request.

^WAIT Indicates that SIL should wait for completion of the I/O request (specified by RSN) before returning control to the caller.

Calling Parameters for Tape Files Only

^LRUA=,addr Address of an array in which Q5CHECK returns one-word descriptions of the LRUs read or written. The array must begin on a word boundary. If LRUA=,addr is omitted, Q5CHECK does not return LRU descriptions.

^LRUL=,n Number of words in the LRU description array; the maximum is 40 words. If LRUL=,n is omitted, 1 is assumed.

Return Parameters

^ERRLEN=,len Error message length in bytes.

^ERRMSG=,msg Error message. The variable msg must be 80 bytes long.

Figure 9-3. Q5CHECK Call Format (Sheet 1 of 2)

Return Parameters

`^IOSTAT=^,sts` Status of the I/O request. SIL can return the following ASCII values:

COM	I/O request completed without errors.
EOF	I/O request completed; end of file encountered.
ERR	I/O request completed with errors.
PEN	I/O request pending; errors encountered.

`^RL=^,rl` Record length (in bytes) actually transferred.

`^STATUS=^,stat` Status code. Possible values: 0 through 199, 204, 206, 207, 303, 1403, 1409, 1411, 1416, 1422, 1472, 1580, 1596, 1604, 1605, 1653, 1708.

Return Parameters for Tape Files Only

`^RLRUL=^,n` Number of entries returned in the LRU description array. If `RLRUL=,n` is omitted, the number of entries is not returned.

`^RLEVEL=^,lev` Level number of the first LRU read (one ASCII character, 0 through 9 or A through F, right justified, zero filled). The level number is returned if the read request reads the LRU terminator. (This parameter does not apply to V tape format.)

Figure 9-3. Q5CHECK Call Format (Sheet 2 of 2)

Q5CHECK is not used for a file connected to a terminal, because no concurrent I/O can be performed for that device type. If a connected file is specified on a Q5CHECK call, Q5CHECK always returns either the COM response and a zero status code or the EOF response and a 1416 status code. It performs no other action.

Q5CHECK issues the GIVE UP CPU (#FF02) resident system message for tapes. Q5CHECK uses the GIVE UP CPU (#52) virtual system message for disk.

Tape I/O Requests

The I, SI, and LB tape formats support the grouping of PRUs into logical record units (LRUs). A tape I/O request can read or write one or more LRUs. Q5CHECK can return a description of each LRU read or written.

Q5CHECK returns each LRU description as a one-word entry in the array specified with the LRUA and LRUL parameters. Figure 9-4 shows the format of each entry. The RLRUL parameter returns the number of entries returned in the array.

As shown in figure 9-4, if the LRU terminator is read or written, the LRU description contains the level number of the LRU. However, if desired, Q5CHECK can return a level number without returning the LRU description. If the I/O request is a request to read one LRU and the request reads the LRU terminator, Q5CHECK returns the level number of the LRU in the variable specified on the RLEVEL parameter.

If a Q5CHECK call specifies tape I/O request parameters for a nontape file, Q5CHECK returns a warning error (status code 1472).

Field	Bits	Content
-	0-3	Unused.
lev	4-7	Level number (0 thru F); applicable only if the end-of-LRU flag is set. (Not applicable to V tape format.)
-	8-10	Reserved.
ed	11	Excess data flag; set to 1 if a read operation fills the buffer before it encounters the end of the LRU.
par	12	Parity flag; set to 1 if a read operation detects a parity error in the LRU.
eor	13	End-of-LRU flag. A read operation sets the flag to 1 when it reads the LRU terminator. A write operation must set the flag to terminate the LRU.
eof	14	End-of-file flag. A read operation sets the flag to 1 when it reads the end of the file or the end of information. A write operation must set the flag to terminate the file.
eoi	15	End-of-information flag. A read operation sets the flag to 1 if it encounters the end of information.
-	16-31	Reserved.
lrsz	32-63	Number of bytes in the LRU. Set for a write operation; returned by a read operation.

Figure 9-4. LRU Description Format

Q5CHECKB - CHECK BLOCK I/O REQUEST STATUS

A Q5CHECKB call (refer to figure 9-5) determines the status of a Q5GETB or Q5PUTB request. The Q5CHECKB call identifies the request by the request serial ordinal (RSN) returned by the Q5GETB or Q5PUTB call.

After determining the request status by examining the buffer operations status word in the FIT, Q5CHECKB, by default, returns control immediately to the caller. However, if the Q5CHECKB call specifies the WAIT parameter, Q5CHECKB suspends program execution until the block I/O request has completed.

A Q5CHECKB call is valid only for Q5GETB or Q5PUTB calls. To check Q5READ and Q5WRITE requests, call the Q5CHECK routine.

Q5CHECKB issues the GIVE UP CPU system message.

<u>Call Format</u>	
CALL Q5CHECKB({ ^LFN=,lfn ^FLUN=,flun } , ^IOSTAT=,sts, optional parameters)
<u>Calling Parameters</u>	
^LFN=,lfn	File name. ^LFN=,lfn must be specified if FLUN=,flun is omitted.
^FLUN=,flun	File logical unit number SIL assigned to the FIT. FLUN=,flun must be specified if LFN=,lfn is omitted.
^NOREDUCE"	Indicates that a file should not be reduced in length at close time.
^RSN=,rsn	Request serial number returned by the Q5GETB or Q5PUTB call. If RSN=,rsn is omitted, SIL checks the last issued Q5GETB or Q5PUTB request.
^WAIT	Indicates that SIL should wait for completion of the I/O request (specified by RSN) before returning control to the caller.
<u>Return Parameters</u>	
^ERRLEN=,len	Error message length in bytes.
^ERRMSG=,msg	Error message. The variable msg must be 80 bytes long.
^IOSTAT=,sts	Status of the I/O request (ASCII, left justified, blank filled). This parameter is required.
COM	I/O request completed without errors.
EOF	I/O request completed; end of file encountered.
ERR	I/O request completed with errors.
PEN	I/O request pending.

Figure 9-5. Q5CHECKB Call Format (Sheet 1 of 2)

Return Parameters

^RL=^,rl	Record length (in bytes) actually transferred.
^STATUS=^,stat	Status code. Possible values: 0 through 199, 204, 206, 207, 303, 1403, 1409, 1411, 1416, 1422, 1580, 1596, 1605, 1653.

Figure 9-5. Q5CHECKB Call Format (Sheet 2 of 2)

Q5CLIOER - CLEAR TAPE I/O ERROR

A Q5CLIOER call (refer to figure 9-6) clears an I/O error status on a tape file.

Once the system returns a fatal tape I/O error (within the message for status code 1476), subsequent I/O requests continue to return the error status until the program calls Q5CLIOER to clear the error.

Q5CLIOER issues the TAPE FUNCTION system message with a subfunction code of clear I/O error condition.

<u>Call Format</u>	
CALL Q5CLIOER({ ^LFN=`,lfn ^FLUN=`,flun } ,optional parameters)
<u>Calling Parameters</u>	
^LFN=`,lfn	File name. LFN=,lfn must be specified if FLUN=,flun is omitted.
^FLUN=`,flun	File logical unit number SIL assigned to the FIT. FLUN=,flun must be specified if LFN=,lfn is omitted.
<u>Return Parameters</u>	
^ERRLEN=`,len	Error message length in bytes (integer).
^ERRMSG=`,msg	Error message. The variable msg must be 80 bytes long.
^STATUS=`,stat	Status code. Possible values: 0 through 199.

Figure 9-6. Q5CLIOER Call Format

Q5CLOSE - CLOSE FILE

Call the Q5CLOSE routine (refer to figure 9-7) to close one or all open files before the end of a task. Normal task completion closes all open files.

The file must be opened with Q5OPEN or Q5GETFIL before Q5CLOSE is called; otherwise, or else Q5CLOSE terminates with fatal error 1406.

Q5CLOSE performs the appropriate I/O completion functions. If the last I/O operation for the file was a write operation, Q5CLOSE writes any data remaining in the output buffers and writes the appropriate end-of-file delimiter for the file format.

If the file was opened for explicit I/O with write access, the modified blocks are written on mass storage. If the file does not have write access, Q5CLOSE returns an error.

If the file is an output file as described in chapter 2 and no other tasks are accessing the file, the file is copied to a front-end system for output.

If the file is connected to a terminal, Q5CLOSE ignores the FC= parameter if specified.

If, while closing all open files, Q5CLOSE encounters an error preventing it from closing a file, it records an error status code and continues closing files; therefore, the status code returned is that of the last error encountered.

The system limits to 110 the number of FITs that can be concurrently associated with a job. Specify the RETFIT parameter on the Q5CLOSE call, or issue a Q5RETFIT call to return a FIT. After SIL returns a file's FIT, it cannot reopen the file until a new FIT is generated.

Call Format

CALL Q5CLOSE({ 'LFN=',lfn
'FLUN=',flun
'*' } ,optional parameters)

Calling Parameters

'LFN=',lfn File name. LFN=,lfn must be specified if * and FLUN=,flun are omitted.

'FLUN=',flun Logical number SIL assigned to the file. FLUN=,flun must be specified if * and LFN=,lfn are omitted.

'*' Indicates that Q5CLOSE should close all open files. * must be specified if LFN=,lfn and FLUN=,flun are omitted.

'FC=',cat Indicates that when SIL closes the file, it should change the category of the file.

 'D' Drop file.

'RETFIT' Indicates that SIL should discard the FIT for the file. If RETFIT is omitted, SIL can reopen the file with this FIT.

Calling Parameters for Mass Storage Files Only

'DUMP' Sets a flag in the FILEI indicating that the file has been dumped. This parameter is allowed only for privileged user calls.

Figure 9-7. Q5CLOSE Call Format (Sheet 1 of 2)

Calling Parameters for Tape Files Only

'LABA=',addr	Address of an array to which Q5CLOSE returns the labels it writes. The array must begin on a word boundary. If LABA=,addr is omitted, Q5CLOSE does not return the labels written.
'LABL=',n	Number of words in the array to which Q5CLOSE returns the labels it writes. The length should be a multiple of 10; if it is not, the length is reduced to a multiple of 10. The valid range is from 10 through 510 words. If LABL=,n is omitted, the length is assumed to be 10 words.
'ULABA=',addr	Address of an array containing user-specified labels to be written after the EOF1 label. The array must begin on a word boundary. If ULABA=,addr is omitted, Q5CLOSE writes no user-specified labels.
'ULABL=',n	Number of words in the array containing the user-specified labels. The length should be a multiple of 10; if it is not, the length is reduced to a multiple of 510. The valid range is from 10 through 510 words. If ULABL=,n is omitted, the length is assumed to be 10 words.

Return Parameters

'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	Error message. The variable msg must be 80 bytes long.
'STATUS=',stat	Status code. Possible values: 0 through 199, 204, 205, 303, 430, 1407, 1408, 1517, 1566, 1685.

Return Parameter for Tape Files Only

'RLABL=',n	Length in bytes of the labels written. The length returned is always a multiple of 80. If RLABL=,n is omitted, Q5CLOSE does not return the length.
------------	--

Figure 9-7. Q5CLOSE Call Format (Sheet 2 of 2)

Tape Label Processing

When a tape file opened for write access and write label processing (LPROC=W) is closed, SIL writes the labels or tapemarks that indicate the end of the file. If the file is an ANSI labeled file, SIL writes an EOF1 label. The contents of the EOF1 label match the contents of the HDR1 label for the file (except for the label identifier and block count).

Write additional end-of-file labels after the EOF1 label. The valid user-specified ANSI labels are EOF2 through EOF9 and UTL. (The ANSI label formats are shown in appendix G of this manual.)

To write additional labels, store the label contents in an array and specify the array address and length with the ULABA and ULABL parameters. If invalid labels in the array are specified, SIL returns a fatal error (status code 1475).

Q5CLOSE returns the contents of the labels it writes if an array to receive the labels using the LABA and LABL parameters is specified. The array should be long enough for all labels written (80 bytes for the EOF1 label, plus the length of any user-specified labels). If the array is too short, SIL returns a warning error (status code 1474).

If the RLABL parameter is specified, SIL returns the length of the labels written.

If tape label parameters for a nontape file are specified, SIL returns a warning error (status code 1473).

If the Q5OPEN call specifies user error processing, the Q5CLOSE call returns status code 1476 if a fatal I/O error occurs.

Q5DEFINE - DEFINE PERMANENT FILE

Call the Q5DEFINE routine (refer to figure 9-8) to create a permanent file. The new permanent file can be a newly created file or an existing local file. File attributes for a newly created file can be specified; existing file attributes cannot.

The action Q5DEFINE performs depends on the file name or number specified.

<u>Q5DEFINE Specified</u>	<u>Result</u>
Permanent file	Error status returned.
Local file	The local file becomes a permanent file attached to the job. The Q5DEFINE call does not change the open or closed status of the file.
Nonexisting file	A new permanent file is created, closed, and attached to the job.
File connected to a terminal	Fatal error returned (status code 1453).

Q5DEFINE cannot create or make permanent a file having a security level higher than the security level of the task.

The new permanent file is not attached to other jobs running under the same user number. If necessary, Q5DEFINE generates a FIT for the new file. If an error occurs during processing, Q5DEFINE does not destroy the FIT.

Privileged Q5DEFINE calls use only the following parameters: LFN= or FLUN=, EXT=, LEN=, NOSEG, FITE=, and FILEL=. Any other parameters specified are ignored. A privileged Q5DEFINE call does not use the values stored in an existing FIT. If a FIT does not exist for the file, it generates one using default values for each field; however, the FIT values used are those you specify via a privileged Q5OPEN call.

Q5DEFINE issues the CREATE FILE system message.

<u>Call Format</u>	
CALL Q5DEFINE($\left\{ \begin{array}{l} \text{`LFN=`,lfn} \\ \text{`FLUN=`,flun} \end{array} \right\}$, optional parameters)
<u>Calling Parameters</u>	
`LFN=`,lfn	File name. LFN=,lfn must be specified if FLUN=,flun is omitted.
`FLUN=`,flun	Logical number SIL assigned to the FIT or local file. FLUN=,flun must be specified if LFN=,lfn is omitted.

Figure 9-8. Q5DEFINE Call Format (Sheet 1 of 4)

Calling Parameters Used Only When Creating a File

<code>^ACS=^,acs</code>	Access permission set (any combination of the following letters without separators). R Read permission W Write permission A Append permission M Modify permission X Execute permission If ACS=,acs is omitted, ^ACS=^,^RWAMX^ is assumed if the file is a new file and if its FIT does not contain access permission information provided by a Q5SETFIT call. Otherwise, the existing access permission set remains unchanged.
<code>^AU=^,blocks</code>	Allocation unit. blocks is an integer specifying the number of 512-word blocks to be used to create or extend the file. The value range is 1 to 65,535. If blocks is not a multiple of DAU, it is rounded up to the next DAU value. If the file is already local, this parameter is ignored.
<code>^BT=^,bt</code>	Blocking type. If BT=,bt is omitted, the file has fixed character count blocking. ^C^ Character count
<code>^CT=^,ct</code>	Front-end communication type. If CT=,ct is omitted, SIL does not change the communication type. ^RHF^ Remote Host Facility ^NRHF^ No Remote Host Facility ^RWF^ Remote Workstation Facility
<code>^EXT=^,ext</code>	File extendability indicator. If EXT=,ext is omitted, the file is extendable. ^Y^ The file is extendable ^N^ The file is not extendable
<code>^FC=^,fc</code>	File category. If FC=,fc is omitted, the file is a user file. ^B^ Batch input file ^U^ User file
<code>^LEN=^,fl</code>	File length in 512-word blocks. If LEN=,fl is omitted, the file is eight 512-word blocks.

Figure 9-8. Q5DEFINE Call Format (Sheet 2 of 4)

Calling Parameters Used Only When Creating a File

'MNR=',mnr	Minimum record length in bytes. For record formats other than F, SIL checks to see that the record is not shorter than this value. SIL does not use this parameter when writing F format records. If MNR=mnr is omitted, SIL assumes that the minimum record length is one byte.
'MXR=',mxr	Maximum record length in bytes. For F records, mxr is the fixed record length. For other record formats, SIL checks to see that the record is not longer than this value. If MXR=,mxr is omitted, SIL assumes that maximum record length is the default set by an installation parameter (released value, 0).
'NOSEG'	Indicates that the initial file space allocated must be contiguous. If NOSEG is omitted, the system can allocate initial file space in multiple segments. To create a file that will always remain contiguous, both the NOSEG and EXT=N parameters must be specified.
'PN=',pn	Six-character identifier of the disk pack on which SIL creates the file. If PN=,pn is omitted, the system assigns mass storage space for the file.
'PC=',pc	Padding character for F format records. If PC=,pc is omitted, SIL pads with the installation-defined padding character (released value, blank).
'RMK=',rmk	Record delimiting character for R format records. If RMK=,rmk is omitted, SIL uses the installation-specified character [released value, ASCII US (#1F)].
'RT=',rt	Record format. If SFO=,D is specified, the only valid record format is F. If RT=,rt is omitted, SIL assumes the installation-defined format (released value, R) for sequential access files and F format for direct access files. 'F' ANSI fixed length 'R' Record mark delimited 'U' Undefined 'W' Control word
'SFO=',fo	File organization. If SFO=,fo is omitted, SIL assumes the installation-defined default organization (released value, sequential access). 'D' Direct access 'S' Sequential access
'SLEV=',sl	Security level (1 through 8, but less than or equal to that of the calling task). If SLEV=,sl is omitted, SIL sets the file security level equal to that of the calling task.

Figure 9-8. Q5DEFINE Call Format (Sheet 3 of 4)

Calling Parameters Used Only When Creating a File

`^TYPE=`,typ File type. If TYPE=,typ is omitted, SIL assumes that the file is a physical data file.

`^PD` Physical data file

`^VC` Virtual code (controllee) file

Calling Parameters for Privileged Users Only

`^FADE=`,aray Name of the 16-word array that is to receive a copy of the file access directory entry for the file if one exists.

`^FITE=`,array Name of the array containing a copy of a file index entry. SIL uses the copy to initialize the file index entry for the file. If FITE=,array is omitted, SIL generates the file index entry.

`^FITEL=`,alen Length (in words) of the array named by the FITE=,array parameter. The system checks to see that this length is the length of a file index entry. If FITEL=,alen is omitted, SIL generates the file index entry.

Return Parameters

`^CONT=`,con Initial contiguity of the mass storage file (ASCII, left-justified, blank filled).

 Y One segment allocated

 N One or two segments allocated

`^DA=`,da Action performed by the Q5DEFINE call (ASCII, left-justified, blank filled).

 N New permanent file created

 Y Local file made permanent

`^ERRLEN=`,len Error message length in bytes (integer).

`^ERRMSG=`,msg Error message. The variable msg must be 80 bytes long.

`^RFLUN=`,rflun File logical unit SIL assigns to the FIT. Number assigned to the file.

`^RPN=`,pn Six-character identifier of the disk pack on which the file resides.

`^STATUS=`,stat Status code. Possible values: 0 through 199, 204, 205, 521, 1453, 1505, 1685, 1688, 1710, 1711, 1716, 1720, 1722, 1724, 1804.

Figure 9-8. Q5DEFINE Call Format (Sheet 4 of 4)

Q5ENDPAR - WRITE PARTITION DELIMITER

Call the Q5ENDPAR routine (refer to figure 9-9) to write a partition delimiter on the file. Q5ENDPAR can write delimiters to mark the end of a record, a group, or a file.

If the user calls Q5ENDPAR to write a group or file delimiter after writing one or more partial W, R, or L records, Q5ENDPAR terminates the record before writing the group or file delimiter. It terminates an R record by writing a record mark character; it terminates a W or L record by writing a last piece control word.

R format uses group and file delimiters only if the record mark character is ASCII US (#1F) or RS (#1E).

Call Q5ENDPAR to write a record or file delimiter on an F or U format file. Q5ENDPAR terminates the partition but does not write a delimiter character on the file. If the Q5ENDPAR is called to write a group delimiter on an F or U format file, Q5ENDPAR writes a file delimiter.

If you call Q5ENDPAR to terminate a partition on an F format file in which the last record is not filled (to RLMAX characters), Q5ENDPAR pads the record (RLMAX characters) and returns a warning (status code 1452).

The Q5PUTN routine automatically writes partition delimiters. When Q5PUTN calls are issued to write a file, Q5ENDPAR calls are not necessary.

If the specified file is connected to a terminal, Q5ENDPAR displays the following output line at the terminal, depending on the value given the PART= variable. The \$ in the output line represents the system special character that prefixes request lines. (For more information, refer to Interactive Request Lines in chapter 3 of this manual).

<u>PART=</u>	<u>Output Line</u>
F	\$EOF
G	\$EOG
R	\$EOR

Call Format

```
CALL Q5ENDPAR( { 'LFN=',lfn } , optional parameters )
                { 'FLUN=',flun }
```

Calling Parameters

'LFN=',lfn	File name. LFN=,lfn must be specified if FLUN=,flun is omitted.
'FLUN=',flun	Logical number SIL assigns to the file. FLUN=,flun must be specified if LFN=,lfn is omitted.
'PART=',part	File partition delimiter. If PART=,part is omitted, SIL writes a record delimiter.
'R'	Record delimiter
'G'	Group delimiter
'F'	File delimiter

Figure 9-9. Q5ENDPAR Call Format (Sheet 1 of 2)

Calling Parameter for Tape Files Only

LEVEL=lev Level number written in the record terminator when writing B records on a tape file (ASCII character, 0 through 9 or A through F). If LEVEL=lev is omitted, 0 is used.

Return Parameters

ERRLEN=len Error message length in bytes (integer).

ERRMSG=msg Error message. The variable msg is 80 bytes long.

STATUS=stat Status code. Possible values: 0 through 199, 303, 1422, 1430, 1444, 1452, 1477, 1478, 1726.

Figure 9-9. Q5ENDPAR Call Format (Sheet 2 of 2)

Tape Partition Delimiters

When writing B format records to a tape file with Q5PUTP calls, the caller can terminate a record or group by calling Q5ENDPAR and specifying PART=R or PART=G. The LEVEL=lev parameter determines the level number written in the record terminator. To terminate a record, the level number must be one of the ASCII characters 0 through 9 or A through E. To terminate a group, specify either PART=G or PART=R and LEVEL=F.

If a Q5ENDPAR call specifies the LEVEL=lev parameter for a nontape file or a V format tape file, Q5ENDPAR returns a warning error (status code 1477).

Q5GENFIT - GENERATE FIT

Call the Q5GENFIT routine (refer to figure 9-10) to generate and initialize a FIT for the specified file name. Q5GENFIT uses parameter specifications and default values to initialize the fields in the FIT.

SIL requires a FIT for each file it reads or writes. The Q5DEFINE and Q5RQUEST calls generate FITs for their files. The system discards those FITs at completion of the task; however, the local and attached permanent files remain attached to the job. A new FIT can be generated for a file attached by a previous task with a Q5GENFIT call or a Q5OPEN call.

By specifying the BUF1= and BUF2= parameters, the virtual address range that is to be the I/O buffer for the file can be assigned. After the task opens the file and executes the first I/O request for the file, the buffer can file data at any time until the file is closed. (SIL flushes the I/O buffers when the file is closed.)

NOTE

Do not assign the same virtual memory space to be an I/O buffer for two files that are open at the same time. Data for one file could overwrite data for the other.

If the specified DT value differs from the device type of an existing local file with the specified name, Q5GENFIT returns a fatal error (status code 1455).

Call Format

CALL Q5GENFIT(`LFN=`,lfn, optional parameters)

Calling Parameters

`LFN=`,lfn	Name of the file for which SIL generates a FIT. This parameter is required.
`ACS=`,acs	Access permission set (any combination of the following letters without separators). The access permission set can be overridden by an access permission set on the subsequent call for the file.
	R Read permission
	W Write permission
	A Append permission
	M Modify permission
	X Execute permission

If ACS=,acs is omitted, the access permission set for the file is not changed.

Figure 9-10. Q5GENFIT Call Format (Sheet 1 of 5)

Calling Parameters

<code>^BT=^,bt</code>	Blocking type C Character count blocking I Internal blocking (tape files only) K Record count blocking (tape files only) If BT=,bt is omitted, character count blocking is assumed.
<code>^BUFL1=^,b11</code>	Length of buffer 1 to 24*N, where N is the number of blocks per page (with N=1, 4, 16 or 128). The parameters are validated when I/O is performed.
<code>^BUFL2=^,b12</code>	Length of buffer 2 to 24*N, where N is the number of blocks per page (with N=1, 4, 16 or 128). The parameters are validated when I/O is performed.
<code>^BUF1=^,b1</code>	Array to be used as data buffer 1. The buffer must be on a page boundary (specified by a LOAD utility parameter). If the buffer is 128 blocks (a large page), it must be on a large page boundary.
<code>^BUF2=^,b2</code>	Array to be used as data buffer 2. The buffer must be on a page boundary (specified by a LOAD utility parameter). If the buffer is 128 blocks (a large page), it must be on a large page boundary.
<code>^CFP=^,cfp</code>	File positioning when the file is closed. If CFP=,cfp is omitted, SIL does not rewind the file. If the file is a direct access file, the parameter specification is ignored; the file is not rewound. ^N^ Do not rewind the file. ^R^ Rewind the file.
<code>^DT=^,dt</code>	Device type. ^MS^ Mass storage ^NT^ Magnetic tape ^TE^ Interactive terminal If DT=,dt is omitted, the DT FIT field value depends on whether a local file with the specified name exists. If the file exists, the DT value indicates the device type of the existing local file; otherwise, the DT value is MS.
<code>^ERL=^,erl</code>	Maximum number of SIL warning errors allowed for the file before SIL aborts the task. If a zero limit is specified or ERL=,erl is omitted, SIL allows an unlimited number of warning errors.

Figure 9-10. Q5GENFIT Call Format (Sheet 2 of 5)

Calling Parameters

<code>^MNR=^,mnr</code>	Minimum record length in bytes. For record formats other than F, SIL checks to ensure that a record is not shorter than this value. SIL does not use this value when writing F-type records. If <code>MNR=,mnr</code> is omitted, SIL assumes that the minimum record length is 1 byte.
<code>^MXR=^,mxr</code>	Maximum record length in bytes. For F record format, <code>mxr</code> is the fixed record length. For other record formats, SIL checks to ensure that the record is not longer than this value. If <code>MXR=,mxr</code> is omitted, SIL assumes that the maximum record length is the default set by an installation parameter.
<code>^OFP=^,ofp</code>	File positioning when the file is opened. If the file is a direct access file, the parameter specification is ignored; the file is not rewound. <code>^N^</code> Do not rewind the file. <code>^P^</code> Position the file after the last block read or written. The TAPETBA and TAPETBL parameters must specify the array containing the tapes table entry copied when the file was last open (tape files only). <code>^R^</code> Rewind the file.
<code>^PC=^,pc</code>	Padding character for F format records. If <code>PC=,pc</code> is omitted, SIL pads with the installation-defined padding character (released value, blank).
<code>^RMK=^,rmk</code>	Record delimiting character for R format records. If <code>RMK=,rmk</code> is omitted, SIL uses the installation-specified character (released value, ASCII US, #1F code).
<code>^RT=^,rt</code>	Record type. <code>^B^</code> System block <code>^F^</code> ANSI fixed length <code>^L^</code> CYBER Record Manager control word <code>^R^</code> Record mark delimited <code>^U^</code> Unstructured <code>^W^</code> Control word

Figure 9-10. Q5GENFIT Call Format (Sheet 3 of 5)

Calling Parameters

`^SFO=`,fo File organization. If SFO=,fo is omitted, SIL assumes the installation-defined default organization (released value, sequential access).

`^D` Direct access

`^S` Sequential access

`^SRF`^,srf Indicates whether SIL must complete an I/O request before returning control to the caller. If SRF=,srf is omitted, SIL returns control to the caller before completing the read or write.

 If the file is a direct access file, the parameter specification is ignored; SIL cannot return control to the caller until after completing the read or write.

`^Y` Suppress overlapped I/O.

`^N` Allow overlapped I/O.

`^WSA=`^,wsa Working storage area used by get and put calls.

`^WSL=`^,wsl Length (in bytes) of the working storage area.

Calling Parameters for Tape Files Only

`^CONV`^ If CONV is specified, the tape data is character data to be converted to and from character codes. If CONV is omitted, the tape data is binary data and no conversion is performed.

`^DEN=`^,den Tape recording density (ASCII, left justified, blank filled).

`^PE` 1600 cpi

`^GE` 6250 cpi

 If DEN=,den is omitted, the installation-defined default density is used (released value, 6250 cpi).

`^LABEL=`^,lbl Tape labels (ASCII, left justified, blank filled).

`^AN` ANSI standard labels

`^NS` Nonstandard labels

`^UL` No labels (unlabeled tape)

 If LABEL=,lbl is omitted, ANSI standard labeling is assumed.

Figure 9-10. Q5GENFIT Call Format (Sheet 4 of 5)

Calling Parameters for Tape Files Only

MPRU=,mpru	MPRU size in bytes for V format tapes (integer). If MPRU=,mpru is omitted, the MPRU size is the installation-defined maximum size (released value, 32768 bytes).
RPB=,rpb	Records per block (integer); valid for K blocking only. If RPB=,rpb is omitted, one record per block is assumed.
TAPETBA=,adr	Address of the array to which each tape I/O request copies the current system tapes table entry for the file. If TAPETBA=,adr is omitted, tape I/O requests do not copy the tapes table entry.
TAPETBL=,n	Number of words in the tapes table entry array (integer). A tapes table entry is 12 words long.
TF=,tf	Tape format (ASCII, left justified, blank filled). I Internal LB Large block SI SCOPE internal V Variable NV Variable format with non-ANSI interchange

If TF=,tf is omitted, the installation-defined default format is used (released value, LB).

Return Parameters

ERRLEN=,len	Error message length in bytes (integer).
ERRMSG=,msg	Error message. The variable msg is 80 bytes long.
RFLUN=,rflun	Number SIL assigned to the file.
STATUS=,stat	Status code. Possible values: 0 through 199, 204, 303, 1455, 1470, 1471, 1472, 1725, 1800.

Figure 9-10. Q5GENFIT Call Format (Sheet 5 of 5)

Tape File FITs

A FIT generated for a tape file must specify NT as its device type with the DT=,dt parameter.

As shown in figure 9-10, certain Q5GENFIT parameters are for tape files only. If a Q5GENFIT call specifies a tape parameter and a device type other than NT, Q5GENFIT returns a warning error (status code 1472).

The values specified for a tape file FIT must not conflict. The valid combinations of tape format, blocking type, and record type parameters are shown in table 9-3.

A Q5GENFIT call should specify the RPB parameter only if it specifies K blocking for the file; otherwise, Q5GENFIT returns a fatal error (status code 1470).

Similarly, a Q5GENFIT call should specify the MPRU parameter only if it specifies V tape format; otherwise, Q5GENFIT returns a fatal error (status code 1471).

Table 9-3. Blocking Type, Tape Format, and Record Type Combinations

Record Type	Blocking Type/Tape Format							
	C/I	C/SI	C/LB	C/V	I/I	I/SI	I/V	K/V
B	x	x	x	x	--	--	--	x
F	--	--	x	x	--	--	--	x
L	--	--	--	--	x	x	x	--
R	--	--	x	x	--	--	--	x
U	x	x	x	x	--	--	--	x
W	--	--	x	x	--	--	--	x

Accessing the Tapes Table Entry

A Q5GENFIT call can specify an array to which tape I/O requests for the file copy the current tapes table entry for the file. The TAPETBA=,adr parameter specifies the array address and the TAPETBL=,n parameter specifies the array length.

The tapes table is the system table that coordinates processing of each requested tape file. A tapes file entry is twelve words long; figure 9-11 shows the entry format.

<u>Word</u>	
1	Reserved for system use
2	Reserved for system use
3	lfn
4	fsn sn
5	Reserved for system use
6	reel vsn
7	bid1 bid2 bid3 bid4
8	bid5 bid6 bid7 bid8
9	bid9 bid10 bid11 bid12
10	fc abc ctfp cbc
11	twre trre stce dtce
12	Reserved for system use

<u>Field</u>	<u>Word</u>	<u>Bits</u>	<u>Content</u>
lfn	3	0-63	File name.
fsn	4	0-15	File sequence number (1 through 9999).
sn	4	16-31	File section number (1 through 9999).
reel	6	0-15	Reel number of the current volume within the multifile set.
vsn	6	16-63	Volume serial number of the current volume.
bid1- bid12	7-9	0-63	Block IDs of the last twelve good PRUs. The block IDs are in the order in which the blocks were read; bid12 is the last good PRU, bid11 is the next-to-last good PRU, and so forth.

Figure 9-11. Tapes Table Entry Format (Sheet 1 of 2)

<u>Field</u>	<u>Word</u>	<u>Bits</u>	<u>Content</u>
fc	10	0-7	Failure code. 1 Nonfatal marginal drive indicator (MDI) 2 Too many erase/ write errors 3 Too many read errors 4 Too many positioning/loop fault errors 5 Fatal marginal drive indicator (MDI)
abc	10	8-31	Absolute PRU count (number of interblock gaps encountered on the tape).
ctfp	10	32-39	Current file position. 0 Within a file 1 End of information 2 End of file 3 End of file and end of information 4 End of LRU 8 Beginning of information
cbc	10	40-63	Current PRU count. It does not include label PRUs.
twre	11	0-15	Total recoverable write errors for the volume.
trre	11	16-31	Total recoverable read errors for the volume.
stce	11	32-47	Total single-track, hardware-corrected errors.
dtce	11	32-47	Total double-track, hardware-corrected errors.

Figure 9-11. Tapes Table Entry Format (Sheet 2 of 2)

Q5GETB - GET A BUFFER RECORD

A Q5GETB call (refer to figure 9-12) gets the next buffer record from the file specified. If the tape file is of B-record type, Q5GETB copies the tape block directly from the file to the specified working storage area; it does not use intermediate I/O buffers.

If the specified file is not a tape file with the B record type, a Q5GETB call is equivalent to a Q5GETN call.

If the call specifies the RSN=,rsn parameter, Q5GETB returns control to you immediately. The task can then perform other processing while the tape I/O request completes processing. To determine whether the I/O request is complete, the program must call the Q5CHECKB routine.

<u>Call Format</u>	
CALL Q5GETB({ ^LFN=`,lfn ^FLUN=`,flun } , optional parameters)
<u>Calling Parameters</u>	
^LFN=`,lfn	File name. LFN=,lfn must be specified if FLUN=,flun is omitted.
^FLUN=`,flun	File logical unit number SIL assigned to the FIT. FLUN=,flun must be specified if LFN=,lfn is omitted.
^WSA=`,wsa	Address of the array to which Q5GETB copies data. If WSA=,wsa is omitted, the working storage area specified in the FIT is used.
^WSL=`,wsl	Length (in bytes) of the working storage area. If WSL=,wsl is omitted, the length specified in the FIT is used.
<u>Return Parameters</u>	
^ERRLEN=`,len	Error message length in bytes (integer).
^ERRMSG=`,msg	Error message. The variable msg must be 80 bytes long.
^RSN=`,rsn	Request serial number. If RSN=,rsn is specified, Q5GETB returns control to you immediately. The task specifies the returned RSN on a Q5CHECKB call to determine whether the Q5GETB request has completed.
	If RSN=,rsn is omitted, the task is not resumed until the Q5GETB request has completed.
^RL=`,rl	Record length (in bytes) actually transferred.
^STATUS=`,stat	Status code. Possible values: 0 through 199.

Figure 9-12. Q5GETB Call Format

Q5GETFIL - OPEN OR CREATE AND OPEN A FILE

A Q5GETFIL call (refer to figure 9-13) either opens an existing file or requests and opens a new local file.

Q5GETFIL cannot request a file having a security level higher than that of the task. An attempt to do so returns a message stating that the file is not found.

If you specify the BUF1=,b1 and BUF2=,b2 parameters, the virtual address range that is to be the I/O buffer for the file can be assigned. The BUF1=,b1 and/or BUF2=,b2 parameters must be specified if you expect to explicitly read or write the opened file. After the task opens the file and executes the first I/O request for the file, the buffer can contain file data at any time until the file is closed. SIL flushes the I/O buffers when the file is closed.

NOTE

Do not assign the same virtual memory space to be an I/O buffer for two files that are open at the same time. Data for one file could overwrite data for the other.

Call Format

CALL Q5GETFIL('LFN=',lfn, optional parameters)

Calling Parameters

'ACS=',acs

Access modes requested.

'R' Read access

'W' Write access

'RW'
or
'WR' Read and write access

'A' Append access

'RA'
or
'AR' Read and append access (mass storage files only)

'M' Modify access (mass storage files only)

'RM'
or
'MR' Read and modify access (mass storage files only)

If ACS=,acs is omitted and the file exists, Q5GETFIL opens the file for the access specified in the FIT for the file, if one exists; otherwise, it opens the file for the access specified when the file was requested, attached, or defined.

If Q5GETFIL requests a new file, it opens the file for read and write access.

Figure 9-13. Q5GETFIL Call Format (Sheet 1 of 8)

Calling Parameters

'AU=',blocks	Allocation unit. The integer number of 512-word blocks to be allocated for the file when it is extended. The value range of blocks is 1 to 65,535. If the file is created and blocks is not a multiple of the DAU for the device in which the first allocation occurs, blocks is rounded up to the next multiple of the DAU.
'BUFL1=',b11	Length of buffer 1 in 512-word blocks. If BUFL1=b11 is omitted, the installation-defined buffer length is used (release value is 3 blocks). The values are 1 to 24*N, where N is the number of blocks per page (with N=1, 4, 16, or 128). The values are validated when I/O is performed.
'BUFL2=',b12	Length of buffer 2 in 512-word blocks. If BUFL2=b12 is omitted, the installation-defined buffer length is used (release value is 3 blocks). The values are 1 to 24*N, where N is the number of blocks per page (with N=1, 4, 16 or 128). The values are validated when I/O is performed.
'BUF1=',b1	Array to be used as data buffer 1. The buffer must be on a page boundary (specified by a LOAD utility parameter). If the buffer is 128 blocks (a large page), it must be on a large page boundary. BUF1 is not required.
'BUF2=',b2	Array to be used as data buffer 2. The buffer must be on a page boundary (specified by a LOAD utility parameter). If the buffer is 128 blocks (a large page), it must be on a large page boundary. Data buffer 2 is not required.
'DT=',dt	Device type on which the file is to reside. DT=,dt is ignored if the file exists. Only DT=,MS is valid for a direct access file. If DT=,dt is omitted and the file does not already exist, the file resides on mass storage. 'MS' Mass storage 'NT' Magnetic tape 'TE' Interactive terminal
'LFN=',lfn	File name. LFN=,lfn is required.
'MNR=',mnr	Minimum record length in bytes. If MNR=,mnr is omitted and Q5GETFIL requests the file, the minimum record length is zero bytes. If MNR=,mnr is omitted and file already exists, its minimum record length is not changed.
'MXR=',mxr	Maximum record length in bytes. If MXR=,mxr is omitted and Q5GETFIL requests the file, no maximum record length is set. If MXR=,mxr is omitted and the file already exists, its maximum record length is not changed.

Figure 9-13. Q5GETFIL Call Format (Sheet 2 of 8)

Calling Parameters

`^NOCOMP^` Indicates that blank compression and expansion should not be performed on the file. If NOCOMP is omitted, blank compression and expansion are performed.

`^OFP=^,ofp` File positioning when the file is opened.

If the file is a direct access file, the parameter specification is ignored; the file is not rewound.

`^N^` Do not rewind the file.

`^R^` Rewind the file.

`^SLEV=^,sl` Security level (1 through 8, but less than or equal to that of the calling task). SLEV=,sl is ignored if the file already exists and Q5GETFIL does not return the file. If SLEV=,sl is omitted and Q5GETFIL requests the file, its security level is that of the calling task.

`^WSA=^,wsa` Working storage area used by get and put I/O calls.

`^WSL=^,wsl` Length (in bytes) of the working storage area.

Calling Parameters for Mass Storage Files Only

`^DC=^,dc` Disposition code. If DC=,dc is omitted and Q5GETFIL requests the file, the system default disposition code is used; if DC=,dc is omitted and the file already exists, its disposition code is not changed.

`^IN^` Batch job input.

`^LR^` Print on a 580-12 printer.

`^LS^` Print on a 580-16 printer.

`^LT^` Print on a 580-20 printer.

`^PF^` Store as a permanent file.

`^PR^` Print on any available line printer.

`^PU^` Punch file.

`^P1^` Print on a 501 printer.

`^P2^` Print on a 512 printer.

`^SC^` Discard file at the end of the task.

`^FC=^,fc` File category. FC=,fc is ignored if the file already exists. If FC=,fc is omitted and Q5GETFIL requests the file, it requests a user file.

`^B^` Batch input file

`^U^` User file

Figure 9-13. Q5GETFIL Call Format (Sheet 3 of 8)

Calling Parameters for Mass Storage Files Only

'IC=',ic	File format. If IC=,ic is omitted and Q5GETFIL requests the file, the file format is the system default; if the file already exists, its file format is not changed. 'AS' 8-bit ASCII code; ANSI carriage control if print file 'BI' Binary 'PA' 8-bit ASCII code; ASCII carriage control if print file
'IMP'	Indicates that the file is to be opened for implicit I/O. The record format in the file's FIT is changed to undefined; the record format in the file's FILEI entry is not changed. If IMP is omitted, the file is opened for explicit I/O.
'LEN=',len	File length in 512-word blocks. If the file already exists, the parameter is ignored. If Q5GETFIL requests the file and LEN=,len is omitted, the file length is eight blocks.
'MODDROP'	Indicates that modified pages of a file opened for implicit I/O are written to the task drop file rather than to the original file. If MODDROP is specified, IMP must also be specified. If MODDROP is omitted, modified pages are written to the original file.
'NOEXT'	Indicates that the file cannot be extended. If NOEXT is omitted, the file can be extended.
'NOSEG'	Indicates that the initial file space allocation cannot be segmented. NOSEG is ignored if the file already exists and is not returned. If NOSEG is omitted and Q5GETFIL requests the file, the initial file space allocation can be segmented.
'PN=',pn	Six-character identifier of a disk pack in the device set on which the file is created. PN=,pn is ignored if the file already exists and is not returned. If PN=,pn is omitted and Q5GETFIL requests the file, the system determines the file residence.
'RETURN'	Indicates that the file is to be returned if it is an existing local mass storage file. If RETURN is omitted, the existing local mass storage file is opened. For more information, refer to the call description.
'SFO=',fo	File organization. If SFO=,fo is omitted, SIL assumes the installation-defined default organization (released value, sequential access). 'D' Direct access 'S' Sequential access

Figure 9-13. Q5GETFIL Call Format (Sheet 4 of 8)

Calling Parameters for Mass Storage Files Only

`^TYPE=^,typ` File type.

- `^PD^` Physical data file
- `^VC^` Virtual code (controllee) file

TYPE=,typ is ignored if the file already exists and is not returned. If TYPE=,typ is omitted and Q5GETFIL requests the file, the file is a physical data file.

Calling Parameters for Mass Storage and Tape Files Only

`^BT=^,bt` Blocking type. If BT=,bt is omitted, character count blocking is assumed.

- `^C^` Character count blocking
- `^I^` Internal blocking
- `^K^` Record count blocking

`^PC=^,pc` ASCII padding character. If PC=,pc is omitted, SIL pads with the installation-defined character (released value, blank).

`^RMK=^,rmk` Record delimiting character for R format records. If RMK=,rmk is omitted, SIL uses the installation-defined default character (released value, ASCII US, #1F code).

`^RT=^,rt` Record format. If SFO=,D is specified, the only valid record format is F. If RT=,rt is omitted, SIL assumes the installation default format (released value, R) for sequential access files and F format for direct access files.

- `^B^` System block (tape files only)
- `^F^` ANSI fixed length
- `^L^` CYBER Record Manager control word (tape files only)
- `^R^` Record mark delimited
- `^U^` Undefined
- `^W^` Control word

Calling Parameters for Privileged Users Only

`^FITE=^,array` Name of the array in which a copy of the file index table for the file is returned. Enter the user number or pool name in the first word of the array and the file name in the second word (refer to the file index entry format in volume 2 of this manual).

`^FITE=^,alen` Length (in words) of the array named by the FITE=,array parameter. The system checks to see that this length is the length of a file index entry. FITE=,alen must be specified if FITE=,array is specified.

Figure 9-13. Q5GETFIL Call Format (Sheet 5 of 8)

Calling Parameters for Privileged Users Only

`^SA=^,sa` Shared access status. If SA=,sa is omitted, the file is opened only if no other tasks have the file open, and no other tasks can open the file until the calling task closes it.

`^Y^` Other tasks can open the file for read access.

`^N^` Other tasks cannot open the file until the file is closed.

Return Parameters

`'ERRLEN=',len` Error message length in bytes (integer).

`'ERRMSG=',msg` Error message. The variable msg must be 80 bytes long.

`'OCAT=',oc` File ownership (ASCII, left-justified, blank filled).

PO Pool file

PR Private file

PU Public file

`'RACS=',acs` Access granted (ASCII, left-justified, blank filled).

R Read access

W Write access

RW
or
WR Read and write access

A Append access

RA
or
AR Read and append access

M Modify access

RM
or
MR Read and modify access

`'RDT=',dt` Device type (ASCII, left-justified, blank filled).

MS Mass storage

NT Magnetic tape

TE Interactive terminal

`'RFLUN=',rflun` Number SIL assigned to this opening of the file.

Figure 9-13. Q5GETFIL Call Format (Sheet 6 of 8)

Return Parameters

'RSLEV=',sl Security level of the file (integer, 1 through 8).

'STATUS=',stat Status code (integer). Possible values: 0 through 199, 303, 1448, 1454, 1459, 1460, 1479, 1483 thru 1486, 1566, 1616, 1710, 1711, 1716, 1720, 1735, 1800, 1801.

Return Parameters for Mass Storage Files Only

'CONT=',cont File contiguity (ASCII, left-justified, blank filled).

 Y The file is contiguous.

 N The file is not contiguous.

'EXT=',ext File extendability (ASCII, left-justified, blank filled).

 Y The file can be extended.

 N The file cannot be extended.

'FSTO=',fsto File segment table ordinal; integer returned only if the file was opened.

'LP=',lp File duration (ASCII, left-justified, blank filled).

 L Local file

 P Permanent file

'NEWFILE=',nfl Indicates whether the opened file is a new file (ASCII, left-justified, blank filled).

 Y Q5GETFIL requested the file.

 N Q5GETFIL opened an existing file.

'RLEN=',len File length in 512-word blocks (integer).

'RPN=',pn Six-character identifier of the pack on which the file resides (ASCII, left-justified, blank filled).

'RTYPE=',typ File type (ASCII, left-justified, blank filled).

 PD Physical data file

 VC Virtual code (controllee) file

Return Parameters for Mass Storage and Tape Files Only

'RBT=',bt Blocking type (ASCII, left-justified, blank filled).

 C Character count blocking

 I Internal blocking (tapes only)

 K Record count blocking (tapes only)

Figure 9-13. Q5GETFIL Call Format (Sheet 7 of 8)

<u>Return Parameters for Tape Files Only</u>											
^RFID=`,fid	File identifier from HDR1 label (17 ASCII characters); applies to ANSI labeled tape files only.										
^RFSN=`,rfsn	File sequence number from HDR1 label (integer); applies to ANSI labeled tape files only.										
^RMPRU=`,mpru	MPRU size in bytes (integer); applies to V format files only.										
^RRPB=`,rpb	Records per block (integer); applies to K blocked files only.										
^RTF=`,fmt	Tape format (ASCII, left justified, blank filled).										
	<table> <tr> <td>I</td> <td>Internal</td> </tr> <tr> <td>LB</td> <td>Large block</td> </tr> <tr> <td>SI</td> <td>SCOPE internal</td> </tr> <tr> <td>V</td> <td>Variable</td> </tr> <tr> <td>NV</td> <td>Non-ANSI, variable</td> </tr> </table>	I	Internal	LB	Large block	SI	SCOPE internal	V	Variable	NV	Non-ANSI, variable
I	Internal										
LB	Large block										
SI	SCOPE internal										
V	Variable										
NV	Non-ANSI, variable										

Figure 9-13. Q5GETFIL Call Format (Sheet 8 of 8)

Mass Storage Files

For mass storage files, the action taken depends on the file specified and the presence or omission of the RETURN parameter on the call. The possible actions are summarized as follows:

<u>Specified File Status</u>	<u>RETURN Omitted</u>	<u>RETURN Specified</u>
Not assigned to job	Requests and opens a new local file	Requests and opens a new local file
A local file	Opens the existing local file	Returns the existing local file and requests and opens a new local file
An attached permanent file	Opens the existing permanent file	Opens the existing permanent file
An attached pool file (except a system pool file)	Opens the existing pool file	Requests and opens a new local file
A public file or system pool file	Requests and opens a new local file	Requests and opens a new local file

Opening a File for Implicit I/O

You can open a mass storage file for implicit I/O and specify that all modified pages of the file be written to the task drop file instead of to the opened file. To do so, specify the IMP and MODDROP parameters. If the IMP parameter is specified, the MODDROP parameter must also be specified; otherwise, Q5GETFIL returns a fatal error (status code 1428).

If the IMP parameter is specified on the Q5GETFIL call to open the file for implicit I/O, then Q5MAPIN must also be called to map in the file.

Files Connected to Terminals

The RETURN parameter has no effect for files connected to a terminal. If the file exists, Q5GETFIL creates a FIT and opens the file. If the file does not exist, Q5GETFIL requests the file, creates its FIT, and opens the file.

Only an interactive task initiated by an interactive execute line (a level 2 controllee) can call Q5GETFIL for a file connected to a terminal. An attempt by a batch controllee or by an interactive controllee at another level returns a fatal error (status code 1459).

If the Q5GETFIL call for a file connected to a terminal specifies mass storage or tape parameters, Q5GETFIL ignores the parameters and returns a warning error (status code 1454).

When you call Q5GETFIL to open an existing file that is only partially available with read-only access, a warning error of 1616 is returned. (Note that this status is returned to you if the status parameter is supplied on a FORTRAN 200 OPEN statement.)

If you try to call Q5GETFIL to create a file greater than the maximum allowed file size, a new fatal status 1710 is returned.

If a specific private device set is selected via the PACK= parameter and allocation is off for all devices in the set with space available, or if all system allocatable packs are off or full, a fatal error status 1711 is returned, as it is when space cannot be allocated.

Tape Files

If the Q5GETFIL call specifies the magnetic tape device type (DT=,NT), Q5GETFIL opens the tape files whether or not the call specifies the RETURN parameter. If a FIT exists for the file, Q5GETFIL uses the existing FIT; otherwise, it generates a FIT, using the information in the file index entry for the file. The file index entry is generated when the file is requested. If the tape file has not previously been requested, Q5GETFIL returns a fatal error (status code 1479).

The specified blocking type (BT) and record type (RT) must not conflict with the tape format specified when the file was requested. The valid combinations of tape format, block type, and record type are shown in table 9-3. If the specified values conflict, Q5GETFIL returns a fatal error (status code 1484).

For ANSI labeled tape files, the specified access mode (ACS) must not conflict with the label processing option specified when the file was requested. If the access mode is read only and labels are to be written, Q5GETFIL returns a fatal error (status code 1480).

If the tape file request specified read only access, the tape volume was mounted without a write ring. If the subsequent Q5GETFIL call for the file specifies write access to the file, Q5GETFIL returns a fatal error (status code 1481).

For V format files, Q5GETFIL checks to ensure that the specified buffers are at least as large as the MPRU size. If a specified buffer is too small, Q5OPEN returns a warning error (status code 1486).

When opening a file in a multfile set, Q5GETFIL searches for an HDR1 label containing the file sequence number and file identifier specified on the file request. If it cannot find the file, it returns a fatal error (status code 1485).

Q5GETFIT - GET FIT FIELD VALUES

Call the Q5GETFIT routine (refer to figure 9-14) to retrieve the contents of specified fields in a file's FIT. SIL copies the FIT field contents to the variable specified by the return parameter. The contents are left justified, blank filled for character data, and right justified, zero filled for numerical data (such as buffer lengths).

<u>Call Format</u>	
CALL Q5GETFIT({ `LFN=` ,lfn `FLUN=` ,flun } ,optional parameters)
<u>Calling Parameters</u>	
`LFN=` ,lfn	File name in the FIT. LFN=,lfn must be specified if FLUN=,flun is omitted.
`FLUN=` ,flun	Logical number SIL assigned to the file and its FIT. FLUN=,flun must be specified if LFN=,lfn is omitted.
<u>Return Parameters</u>	
`ACS=` ,acs	Access permission set. SIL returns a character string composed of the following letters: R Read permission W Write permission X Execute permission A Append permission M Modify permission
`BUFL1=` ,b1	Length of buffer 1 in 512-word blocks.
`BUFL2=` ,b12	Length of buffer 2 in 512-word blocks.
`BUF1=` ,b1	Array to be used as data buffer 1.
`BUF2=` ,b2	Array to be used as data buffer 2.
`BN=` ,bn	Block ordinal of the last disk block SIL has accessed.
`BT=` ,bt	Blocking type (ASCII, left justified, blank filled). C Character count blocking I Internal blocking K Record count blocking
`CBO=` ,cbo	Current byte offset within the data buffer. It is used to determine the buffer space remaining for get and put I/O operations.

Figure 9-14. Q5GETFIT Call Format (Sheet 1 of 5)

Return Parameters

'CLTYP=',ctp	Close type. The type of close operation performed corresponds to the type of open operation performed. SIL can return the following ASCII values: R Nonprivileged P Privileged Ul Privileged system task
'DT=',dt	Device type of the file (ASCII, left-justified, blank filled). MS Mass storage NT Magnetic tape TE Interactive terminal
'ECT=',ect	Number of SIL warning errors issued for the file.
'ERL=',erl	Maximum number of SIL warning errors allowed for the file before SIL aborts the task.
'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	Error message. The variable msg must be 80 bytes.
'ES=',es	Last SIL error code for the file.
'FNF=',fnf	Indicates whether the last SIL error for the file was fatal. SIL can return the following ASCII values: Y The last error was fatal. N The last error was nonfatal.
'HBYTE=',hbyte	Highest available byte in the file.
'LEN=',len	Length in bytes of the data transferred in the last I/O operation on this file.
'LFP=',lfp	Logical file position (ASCII, left-justified, blank filled). BOI Beginning of information BOV Beginning of volume BOF Beginning of logical file MPF Within a file (positioned somewhere between BOI and EOI) MLR Within a logical record

Figure 9-14. Q5GETFIT Call Format (Sheet 2 of 5)

Return Parameters

EOR	End of logical record
EOG	End of group (R and W formats only)
EOF	End of logical file
EOV	End of volume
EOI	End of information
'LLOP=',lop	Last logical operation performed on the file (two-digit number value). The possible values and their meanings are listed in the FIT field description in appendix D of this manual.
'MNR=',mnr	Minimum record length in bytes. For record types other than F, SIL checks to ensure that the record is not shorter than this value. SIL does not use this parameter when writing F format records.
'MXR=',mxr	Maximum record length in bytes. For F format records, mxr is the fixed record length. For other record types, SIL checks to ensure that the record is not longer than this value.
'NBYTE=',nbyte	Next available byte in the file.
'OCS=',ocs	Indicates whether file is open or closed. SIL can return the following ASCII values: N The file has never been opened. O The file is open for explicit I/O. I The file is open for implicit I/O. C The file is closed.
'OFF=',ofp	File positioning when the file is opened (ASCII, left-justified, blank filled). N Do not rewind the file. P Position the file after the last block read or written, as indicated by information in the tapes table entry copied when the file was last open (tape files only). R Rewind the file.
'PC=',pc	Padding character for F format records.
'PEF=',pef	Indicates whether a parity error occurred during file I/O. SIL can return the following ASCII values: Y A parity error occurred. N No parity error occurred.

Figure 9-14. Q5GETFIT Call Format (Sheet 3 of 5)

Return Parameters

'PTL=',ptl	Partial transfer length in bytes (the amount of data transferred by the last Q5GETP or Q5PUTP call).
'RC=',rc	Number of last full record read or written.
'RFLUN=',rflun	Number SIL assigned to the file.
'RL=',rl	Record length (in bytes) actually transferred.
'RLFN=',lfn	ASCII file name.
'RMK=',rmk	Record delimiting character for R format records.
'RT=',rt	Record type (ASCII, left-justified, blank filled). B System block F ANSI fixed length L CYBER Record Manager control word R Record mark delimited U Undefined W Control word
'SFO=',fo	File organization. SIL can return the following ASCII values: D Direct access S Sequential access
'SRF=',srf	Indicates whether SIL must complete an I/O request before returning control to you. SIL can return the following ASCII values: Y I/O overlap suppressed. N I/O overlap allowed.
'STATUS=',stat	Status code. Possible values: 0 through 199, 303.
'UNIT=',dn	Logical device number for the unit on which the file resides (used by operating system).
'WSA=',wsa	Working storage area used by get and put I/O calls.

Figure 9-14. Q5GETFIT Call Format (Sheet 4 of 5)

Return Parameters

'WSL=',wsl Length (in bytes) of the working storage area.

'WPF=',wfp Indicates whether the last I/O operation was a write operation. SIL can return the following ASCII values:

Y	The last operation was a write.
N	The last operation was not a write.

Return Parameters for Tape Files Only

'CONV=',cvt Data conversion option (ASCII, left-justified, blank filled).

Y	Tape data is character data to be converted to and from character codes.
N	Tape data is binary data, and no conversion is performed.

'DEN=',den Recording density (ASCII, left-justified, blank filled).

PE	1600 cpi
GE	6250 cpi

'FID=',fid File identifier from HDR1 label (17 ASCII characters); applies to ANSI labeled tape files only.

'FSN=',rfsn File sequence number from HDR1 label (integer); applies to ANSI labeled tape files only.

'IOER=',ioer Error code for last tape I/O error (integer). Refer to the tape error list in table B-4 in appendix B of this manual.

'MPRU=',mpru MPRU size in bytes (integer); applies to V format files only.

'RPB=',rpb Records per block (integer); applies to K blocked files only.

'TAPETBA=',adr Address of the array to which each tape I/O request copies the current system tapes table entry for the file. If the address is zero, tape I/O requests do not copy the tapes table entry.

'TAPETBL=',n Number of words in the tapes table entry array (integer). A tapes table entry is twelve words long.

'TF=',tf Tape format (ASCII, left-justified, blank filled).

I	Internal
LB	Large block
SI	SCOPE internal
V	Variable
NV	Non-ANSII, variable

Figure 9-14. Q5GETFIT Call Format (Sheet 5 of 5)

Q5GETN - READ PARTITION

Call the Q5GETN routine (refer to figure 9-15) to transfer a logical partition (record, group, or file) of data into the working storage area.

For sequential access, SIL skips to the beginning of the next partition before transferring data. For direct access, SIL reads from the starting address of the specified record.

If the requested direct access record does not exist, Q5GETN returns a fatal error (status code 1434); the data in the working storage area remains unchanged.

SIL transfers the data from a physical I/O buffer specified in the FIT. (Two buffers can be specified and used alternately.) SIL automatically reads data from the file when a buffer is empty.

If the data in the partition exceeds the working storage area, SIL truncates the data, skips to the next partition, and returns a warning error (status code 1436).

Q5GETN transfers partition delimiters of partitions with levels lower than those specified. For example, if the specified partition is a file, Q5GETN transfers record and group delimiters to the working storage area.

If a working storage area is not specified on the Q5GETN call, SIL uses the working storage area specified in the file's FIT.

SIL recognizes logical partitions as described under Logical Record Formats in chapter 2 of this manual.

Q5GETN expands compressed strings of blank characters within R format partitions unless the NOCOMP parameter is specified on the Q5OPEN call for the file.

For more information on blank compression, refer to Record Mark Delimited (R) Record Format in chapter 2 of this manual.

The file must be open for read access. If it is not, Q5GETN returns a fatal error (status code 1726).

Tape Files

In general, getting a tape record is the same as getting a mass storage record. However, besides the F, R, U, and W record types, tape files can also use the B and L record types.

When getting an L record, Q5GETN strips off the block control word and record control words before copying the data to the working storage area.

A B record is an LRU. If the tape is written using I, SI, or LB tape format, each LRU is terminated by a short or zero-length PRU. The LRU terminators are not copied to the working storage area.

An LRU terminator contains a level number, in the range 0 through 9 and A through F. If the level number is F, the LRU terminator terminates a group.

If the RLEVEL parameter is specified on the Q5GETN call when an LRU terminator is read, Q5GETN returns the level number in the specified variable. The RLEVEL parameter is valid only when reading an I, SI, or LB format tape file; if the RLEVEL parameter is specified for a nontape file or a V format tape file, Q5GETN returns a warning error (status code 1477).

Call Format

CALL Q5GETN({ 'LFN=',lfn }
{ 'FLUN=',flun } ,optional parameters)

Calling Parameters

'LFN=',lfn File name. LFN=,lfn must be specified if FLUN=,flun is omitted.

'FLUN=',flun Logical number SIL assigned to the file. FLUN=,flun must be specified if LFN=,lfn is omitted.

'PART=',part File partition to be transferred. If PART=,part is omitted, Q5GETN transfers a record.

 'R' Record

 'G' Group (R, W, and L formats only)

 'F' File

'REC=',n Record number (integer greater than zero). Ignored for a sequential access file. If REC=,n is omitted for a direct access file, the value in the RC FIT field plus 1 is used.

'WSA=',wsa Working storage area. Specifying this parameter overwrites the corresponding FIT field with the specified value. If WSA=,wsa is omitted, SIL uses the working storage area specified in the file's FIT.

'WSL=',wsl Length of working storage area in bytes. Specifying this parameter overwrites the corresponding FIT field with the specified value. If WSL=,wsl is omitted, SIL uses the working storage length specified in the file's FIT.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'RL=',rl Record length (in bytes) actually transferred.

'STATUS=',stat Status code. Possible values: 0 through 199, 203, 253, 254, 303, 1401, 1405, 1416, 1430, 1432, 1434 through 1438, 1441, 1444, 1445, 1468, 1477, 1726, 1810.

Return Parameter for Tape Files Only

'RLEVEL=',lvl Level number in the LRU terminator read (ASCII character, binary, right-justified, zero filled); applicable to I, SI, and LB tape formats only. Its possible values are 0 through 9, A through F.

Figure 9-15. Q5GETN Call Format

Q5GETP - READ PARTIAL PARTITION

Call the Q5GETP routine (refer to figure 9-16) to transfer part of a logical partition (record, group, or file) of data into the working storage area.

Q5GETP copies the data from a physical I/O buffer that SIL maintains. SIL automatically reads data into the buffers as needed.

Q5GETP transfers partition delimiters of partitions with levels lower than those specified. For example, if the specified partition is a file, Q5GETP transfers record and group delimiters to the working storage area.

Check for the appropriate end-of-partition status code to determine when SIL has read the end-of-partition delimiter. The following are the end-of-partition codes.

<u>Code</u>	<u>Meaning</u>
1434	End of information for sequential access files. Record not found for direct access files.
1440	End of record.
1441	End of group.
1416	End of file.

If the SKIP parameter is specified on the Q5GETP call, SIL skips to the beginning of the next partition before transferring data.

<u>Call Format</u>	
CALL Q5GETP({ ^LFN=,lfn ^FLUN=,flun } ,optional parameters)
<u>Calling Parameters</u>	
^LFN=,lfn	File name. LFN=,lfn must be specified if FLUN=,flun is omitted.
^FLUN=,flun	Logical number SIL assigned to the file. FLUN=,flun must be specified if LFN=,lfn is omitted.
^PART=,part	File partition to be transferred. If PART=,part is omitted, Q5GETP transfers a record.
^R	Record
^G	Group (R, W, and L formats only)
^F	File

Figure 9-16. Q5GETP Call Format (Sheet 1 of 2)

Calling Parameters

REC=,n	Record number (integer greater than zero). Ignored for a sequential access file. If REC=,n is omitted for a direct access file, the record number in the RC FIT field plus 1 is used.
SKIP	Indicates that SIL must transfer data from the beginning of a partition. If the file is positioned within a partition, SIL skips to the next partition delimiter. If SKIP is omitted, SIL transfers data from the current file location.
WSA=,wsa	Working storage area. Specifying this parameter overwrites the corresponding FIT field with the specified value. If WSA=,wsa is omitted, SIL uses the working storage area specified in the file's FIT.
WSL=,wsl	Length of working storage area in bytes. Specifying this parameter overwrites the corresponding FIT field with the specified value. If WSL=,wsl is omitted, SIL uses the working storage length specified in the file's FIT.

Return Parameters

ERRLEN=,len	Error message length in bytes (integer).
ERRMSG=,msg	Error message. The variable msg must be 80 bytes long.
RL=,rl	Length (in bytes) of data transferred. This value is the working storage area length unless SIL encountered an error or a partition delimiter.
STATUS=,stat	Status code. Possible values: 0 through 199, 203, 253, 254, 303, 1401, 1405, 1416, 1430, 1432, 1434, 1435, 1437, 1438, 1440, 1441, 1444, 1445, 1456, 1468, 1726, 1810.

Return Parameter for Tape Files Only

RLEVEL=,lvl	Level number in the LRU terminator read (binary, zero filled); applicable to I, SI, and LB tape formats only.
-------------	---

Figure 9-16. Q5GETP Call Format (Sheet 2 of 2)

Q5GETP expands compressed R-type records unless the NOCOMP parameter is specified on the Q5OPEN call for the file. Blank compression is described under Record Mark Delimited (R) Record Format in chapter 2 of this manual.

If a working storage area is not specified on the Q5GETP call, SIL uses the working storage area specified in the file's FIT.

Q5GETP expands compressed strings of blank characters within R format partitions unless the NOCOMP parameter is specified on the Q5OPEN call for the file. For more information on blank compression, refer to Record Mark Delimited (R) Record Format in chapter 2 of this manual.

The file must be open for read access. If it is not, Q5GETP returns a fatal error (status code 1726).

Specifying a file connected to a terminal on a Q5GETP call returns a fatal error (status code 1456).

Tape Files

In general, getting a partial tape record is the same as getting a partial mass storage record. However, besides the F, R, U, and W record types, tape files can also use the B and L record types.

When getting an L record, Q5GETP strips off the block control word and record control words before copying the data to the working storage area.

A B record is an LRU. If the tape is written using I, SI, or LB tape format, each LRU is terminated by a short or zero-length PRU. The LRU terminators are not copied to the working storage area.

An LRU terminator contains a level number, in the range 0 through 9 and A through F. If the level number is F, the LRU terminator terminates a group.

If the RLEVEL parameter is specified on the Q5GETP call when an LRU terminator is read, Q5GETP returns the level number in the specified variable. The RLEVEL parameter is valid only when reading an I, SI, or LB format tape file; if the RLEVEL parameter is specified for a nontape file or a V format tape file, Q5GETP returns a warning error (status code 1477).

Q5GIVE - GIVE FILE OWNERSHIP

The Q5GIVE routine (refer to figure 9-17) transfers ownership of a private file. If the CUSER parameter is not used, the file must be closed and must be either a local mass storage file or an attached permanent file. If the CUSER parameter is used, the file must be an unattached permanent file. Only privileged tasks are allowed to use the CUSER parameter.

A file owner may give an attached private file. Only the pool boss can give a pool file.

Nonprivileged users can give a file to another user or to a pool. To do so, the nonprivileged user must specify the name or number of the file and either the user number or the pool to which the file is given.

Privileged users can also give a file to the public file list. To do so, the user specifies the name of the file and the PUBLIC parameter.

When you give a private file to another user, Q5GIVE immediately changes the user number that owns the file. However, the account identifier associated with the file does not change until the file is referenced by its new owner. The system accounting tables then indicate the total time that the original account owned the file.

After ownership transfer, the file is an unattached permanent file belonging to the specified owner. The file is no longer attached to the job that issued the Q5GIVE call.

Q5GIVE cannot give a file to a user number whose maximum security level is less than the security level of the file. An attempt to do so returns a fatal error (status code 1525).

Q5GIVE cannot change the ownership of a file connected to a terminal. An attempt to do so returns a fatal error (status code 1457).

You can specify a variable rate accounting factor for the file, using the VRI parameter. The system provides variable rate accounting for public utilities and application packages that are not to be charged the full rate. The VRI parameter specifies an index into the variable rate table, T_VRF. You should consult a systems analyst for the appropriate index value.

Q5GIVE uses the GIVE FILE system message.

Call Format

```
CALL Q5GIVE( { ^LFN=^,lfn } , { ^USER=^,un }
              { ^FLUN=^,flun } , { ^POOL=^,pool } , optional parameters)
              ^PUBLIC^
```

Calling Parameters

^LFN=^,lfn	File name. LFN=,lfn must be specified if FLUN=,rflun is omitted.
^FLUN=^,rflun	Number SIL assigned to the file. FLUN=,rflun must be specified if LFN=,lfn is omitted.
^USER=^,un	User number to which SIL gives the file (if POOL=, or PUBLIC is not specified). The user number must be an ASCII string, left-justified, blank-filled.

Figure 9-17. Q5GIVE Call Format (Sheet 1 of 3)

Calling Parameters

`^ACS=^,acs` New access permission set (any combination of the following letters without separators).

R	Read permission
W	Write permission
X	Execute permission
A	Append permission
M	Modify permission

The effect of the ACS parameter and its defaults are shown in table 9-4.

`^VRI=^,vri` Variable rate index (0 through 255). If `VRI=,vri` is omitted, `SIL` sets the variable rate index to zero.

Calling Parameters for Pool Files Only

`^POOL=^,pool` Name of the pool to which `SIL` transfers file ownership. You must specify either `USER=,un`, `POOL=,pool`, or `PUBLIC`.

Calling Parameters for Privileged Callers

`^CUSER=^,cuser` Current owner's user number (ASCII, left justified, blank filled). When specified, the file specified by LFN or FLUN that has this user number is given to the user number specified by `USER`.

`^FIJDN=^,fjdn` FILEI job descriptor number (binary). If specified when giving a file to the output queue, VSOS will attempt to assign the value of `fjdn` to the `FI_JDN` field of the file's FILEI. The value of `fjdn` must either be zero or the same as the caller's in the range 1 through 2047. This parameter is mutually exclusive with the `RJDN`, `POOL`, `CUSER`, and `PUBLIC` parameters.

`^RJDN=^,rjdn` A return parameter. If specified when giving a file to the output queue, the value that VSOS assigned to the `FI_JDN` field of the FILEI is returned. The value of `rjdn` must be in the range 1 through 2047. If `fjdn` is zero, the caller's `jdn` is assigned to the `FI_JDN` field. This parameter is mutually exclusive with the `FIJDN`, `POOL`, `CUSER`, and `PUBLIC` parameters.

`^PUBLIC^` Indicates that file ownership is transferred to the public file list. You must specify either `USER=,un`, `POOL=,pool`, or `PUBLIC`.

Figure 9-17. Q5GIVE Call Format (Sheet 2 of 3)

Calling Parameters for Privileged Callers

`^FLAGS=^,flg` Indicates requested SIL action. If `FLAGS=,flg` is omitted, no action is taken.
`^COU^` Clear the originating user field in the file index table.
`^SPB^` Set the file's privileged bit to allow the file, if executed, to issue privileged calls. (The file's controllers do not have privileged status.)
`^CSPB^` Perform the actions of both COU and SPB.
`^FIOUSER=^,ouser` FILEI originating user field (ASCII, left justified, blank filled). If specified, VSOS will assign the binary value of ouser to the OUSER field in the file's FILEI. This parameter is mutually exclusive with the `^FLAGS=^,^COU^` and `^FLAGS=^,^CSPB^` parameters.

Return Parameters

`^ERRLEN=^,len` Error message length in bytes (integer).
`^ERRMSG=^,msg` Error message. The variable msg must be 80 bytes long.
`^STATUS=^,stat` Status code. Possible values: 0 through 199, 204, 205, 303, 0510, 0512, 0513, 1425, 1457, 1525, 1644, 1680, 1682, 1683, 1685 through 1691, 1709, 1722, 1812, 1814, 1815, 1816.

Figure 9-17. Q5GIVE Call Format (Sheet 3 of 3)

Table 9-4. Q5GIVE Default Access Permission Sets

Old File Ownership	New File Ownership	Default Access Permission Sets
Private	Private	The old owner's access permission set becomes the new owner's access permission set.
Private	Pool	The old owner's access permission set becomes the general access permission set for all pool members and the pool boss.
Private	Public	The general access permission set contains read and execute permissions.
Pool	Private	The old pool boss access permission set becomes the new owners' access permission set.
Pool	Pool	The old pool boss access permission set becomes the new pool boss access permission set; the old general access permission set becomes the new general access permission set.
Pool	Public	The general access permission set contains read and execute permissions.

Q5LABEL - REQUEST FILE FROM MULTIFILE SET

A Q5LABEL call (refer to figure 9-18) specifies the contents of an ANSI standard HDR1 label for a tape file. A Q5LABEL call is applicable only if the Q5REQUEST call does not specify the contents of the HDR1 label for the file.

NOTE

A Q5LABEL call performs the same function as a LABEL control statement. If a Q5LABEL call specifies a file previously specified on a LABEL statement, Q5LABEL returns a fatal error (status code 1626).

The HDR1 label specifications are stored in the file index entry of the file. The information in the file index entry is referenced when the file is opened.

If the tape file is opened for write access, the specified values are written in the new HDR1 label. If the tape file is opened for read access, the specified HDR1 values are compared with the values in the existing HDR1 label.

Before a Q5LABEL statement is processed, a tape file request must specify the file name. (A tape file request can be either a REQUEST control statement or a Q5RREQUEST call.) Unless the operator is to mount an unlabeled tape volume for the file, the tape file request also specifies the tape volumes for the file.

The tape file request and the Q5LABEL call share several data format parameters. If the same parameter is specified on the tape file request and the Q5LABEL call, the value on the Q5LABEL call overrides the value on the tape file request.

A LABEL statement or Q5LABEL call is required to read or write a file that is a member of a multifile set.

<u>Call Format</u>	
CALL Q5LABEL(^LFN=^,lfn,optional parameters)	
<u>Calling Parameters</u>	
^LFN=^,lfn	File name by which the tape file is referenced in the job (one to eight characters). This parameter is required.
^FA=^,x	File accessibility character. If the HDR1 label is read, the specified character must match the accessibility character in the label. If the HDR1 label is written, the specified character is written in the label. If FA=,x is omitted, the default character is determined by an installation parameter (released value, blank).

Figure 9-18. Q5LABEL Call Format (Sheet 1 of 4)

Calling Parameters

<code>^FID=^,fid</code>	File identifier (1 to 17 characters). If FID=,fid is omitted and the HDR1 label is written, the file identifier field in the HDR1 label is all blanks. If FID=,fid is omitted and the HDR1 label is read, the contents of the file identifier field is not checked.
<code>^OFA=^,x</code>	Original file accessibility character. If the HDR1 label is to be overwritten, the specified character must match the accessibility character in the existing label. If OFA=,x is omitted, the default character is determined by an installation parameter (released value, blank).
<code>^RP=^,rp</code>	Retention period in days. The retention period is added to the creation date to determine the expiration date for the file. If RP=,rp is omitted, the default value is 30 days.

NOTE

When a new file is added to a multifile set, its expiration date cannot be later than the expiration date of the first file in the multifile set.

Calling Parameters Used to Override Tape File Request Values

<code>^ACS=^,acs</code>	Data access requested. <code>^R^</code> Read access <code>^W^</code> Write access <code>^RW^</code> or <code>^WR^</code> Read and write access If ACS=,acs is omitted, the access specified on the tape file request is used (default, R).
<code>^BT=^,bt</code>	Blocking type. <code>^I^</code> Internal <code>^C^</code> Character count <code>^K^</code> Record count If BT=,bt is omitted, the blocking type specified on the tape file request is used (default, C).

Figure 9-18. Q5LABEL Call Format (Sheet 2 of 4)

Calling Parameters Used to Override Tape File Request Values

^CONVERT^	Data conversion option. If CONVERT is specified, tape data is read and written as character codes. If CONVERT is omitted, data conversion depends on whether CONVERT is specified on the tape file request.
^LPROC=^,lp	Label processing option. ^R^ Read existing labels (verify existing HDR1 label). ^W^ Write new labels. If LPROC=,lp is omitted, the label processing value specified on the tape file request is used (default: R for read or read and write access, W for write access).
^MXR=^,maxr	Maximum record length in bytes. If MXR=,maxr is omitted, the maximum record length specified on the tape file request is used (default, 0).
^MNR=^,minr	Minimum record length in bytes. If MNR=,minr is omitted, the minimum record length specified on the tape file request is used (default, 1).
^MPRU=^,mpru	MPRU size in bytes; used only if the file uses the V tape format. If MPRU=,mpru is omitted, the MPRU size specified on the tape file request is used (default, 32768).
^PC=^,x	Padding character. If PC=,x is omitted, the padding character specified on the tape file request is used (default, blank).
^RMK=^,x	Character used as the record delimiter for R format records. If RMK=,x is omitted, the record mark delimiter specified on the tape file request is used [default, ASCII US (code #1F)].
^RPB=^,rpb	Records per block; used only for the K blocking type. If RPB=,rpb is omitted, the records per block value specified on the tape file request is used (default, 1).
^RT=^,rt	Record type. ^B^ System block ^F^ ANSI fixed length ^L^ CYBER Record Manager (CRM) control word ^R^ Record mark delimited ^U^ Undefined ^W^ Control word delimited If RT=rt is omitted, the record type specified on the tape file request is used (default, R).

Figure 9-18. Q5LABEL Call Format (Sheet 3 of 4)

Calling Parameters for Multifile Sets Only

^MFN=^,mfn Multifile set name (one to eight characters). The multifile set name must be specified on a previous tape file request. If MFN=,mfn is omitted, the multifile set name is assumed to be the same as the file name.

^FSN=^,fsn File sequence number (1 to 9999). If FSN=9999, a new file is added to an existing multifile set. If FSN=0000, file sequence number 0001 is used.

If FSN=,fsn is omitted, the file is identified by its file identifier as specified on the FID parameter. If both FSN=,fsn and FID=,fid are omitted, the first file in the set is opened.

Return Parameters

^ERRLEN=^,len Error message length in bytes (integer).

^ERRMSG=^,msg Error message. The variable msg must be 80 bytes long.

^RFLUN=^,rflun File logical unit number (integer). Subsequent calls can reference the file by this number instead of by file name.

^STATUS=^,stat Status code. Possible values: 0 through 199, 1624, 1625, 1627.

Return Parameters for Tape Files Only

^RACS=^,racs Access permission set assigned to the tape file by the system (ASCII, left justified, blank filled).

 R Read permission.

 W Write permission.

 R/W Read and Write permission.

Figure 9-18. Q5LABEL Call Format (Sheet 4 of 4)

Multifile Sets

A multifile set is a set of tape files. The set can reside on one or more tape volumes. Each file begins with an HDR1 label and ends with an EOF1 label.

The tape file request specifies the name of the multifile set and the tape volumes that belong to the set. A LABEL statement or Q5LABEL call must specify each file in the set that is to be read or written in the job.

The file name specified on the Q5LABEL call is the file name specified on the Q5OPEN call for the file. The file name specified on the Q5LABEL call is not stored with the file and is discarded when the file is returned.

A file within a multifile set is identified by its file sequence number and its file identifier. The file sequence number and file identifier are written in its HDR1 label when the file is written.

The data format parameter values on the tape file request for the multifile set apply to all files in the set unless the parameter is also specified on the label statement or call for the file.

The Q5LABEL call can specify the same label parameters for a file in a multifile set as for any tape file.

Writing a Multifile Set

To write a multifile set, specify 1 as the file sequence number for the first file of the set and 9999 as the file sequence number for all subsequent files. If the specified file sequence number is 9999, the actual file sequence number written is the next number in sequence for the set.

The Q5LABEL calls for the files in a multifile set can be in any order. No error is returned if a file specified on a Q5LABEL call is not opened.

For example, suppose a program contains the following calls:

```
CALL Q5REQUEST('MFN=', 'FILESET', 'DT=', 'NT',
+ 'VSN=', 'VSNLIST', 'ACS=', 'W', 'RT=', 'B')
CALL Q5LABEL('LFN=', 'X', 'MFN=', 'FILESET',
+ 'FID=', 'FILE1', 'FSN=', 1)
CALL Q5LABEL('LFN=', 'Y', 'MFN=', 'FILESET',
+ 'FID=', 'FILE2', 'FSN=', 9999)
CALL Q5OPEN('LFN=', 'X')
CALL Q5PUTB('LFN=', 'X', 'WSA=', 'DATA1', 'WSL=', 512)
CALL Q5CLOSE('LFN=', 'X')
CALL Q5OPEN('LFN=', 'Y')
CALL Q5PUTB('LFN=', 'Y', 'WSA=', 'DATA2', 'WSL=', 512)
CALL Q5CLOSE('LFN=', 'Y')
```

The Q5REQUEST call specifies FILESET as the multifile set name for files written on the tape volumes specified in the array VSNLIST. The Q5LABEL calls specify HDR1 labels for files in the multifile set.

The first Q5OPEN call writes the HDR1 label specified on the first Q5LABEL call. The file sequence number in the label is 1, and the file identifier is FILE1. The first Q5PUTB call writes a tape block on the file.

The second Q5OPEN call writes the HDR1 label specified on the second Q5LABEL call. The file sequence number in the label is 2 (the next number in sequence for the set), and the file identifier is FILE3. The second Q5PUTB call writes a tape block to the second file in the multifile set.

Reading a Multifile Set

To read a file in a multifile set, specify its file sequence number on its Q5LABEL call. The file sequence number specifies the HDR1 label read when the file is opened.

If the file sequence number of the file is unknown, but the file identifier is known, specify the file identifier and omit the file sequence number on the Q5LABEL call. If a file identifier is specified, the HDR1 labels in the multifile set are searched until the label containing the specified file identifier is found.

If the HDR1 label of the file contains a nonblank accessibility character, the Q5LABEL call must specify the accessibility character on the OFA=,x parameter.

For example, suppose a program contains the following calls to read a block from the second file in a multifile set named FILESET.

```
CALL Q5RQUEST('LFN=', 'FILESET', 'DT=', 'NT',  
+ 'VSN=', 'VSNLIST', 'RT=', 'B')  
CALL Q5LABEL('LFN=', 'X', 'MFN=', 'FILESET',  
+ 'FID=', 'SECOND', 'FSN=', 2)  
CALL Q5OPEN('LFN=', 'X')  
CALL Q5GETB('LFN=', 'X', 'WSA=', 'DATA1', 'WSL=', 512)  
CALL Q5CLOSE('LFN=', 'X')
```

The Q5RQUEST call specifies the multifile set name and the tape volumes to be read. The Q5LABEL call specifies a local file name for the file, the multifile set name, the file identifier, and the file sequence number. The Q5OPEN call opens file X. To open file X, the Q5OPEN call reads the second HDR1 label in the multifile set and checks to see that its file identifier is SECOND and that its file accessibility character is blank. The Q5GETB call reads a tape block and the Q5CLOSE call closes the file.

Rewriting Files in a Multifile Set

Files in a multifile set can be rewritten. However, the last file written is always the last file in the set. So, when rewriting a file in a multifile set, all subsequent files to be kept in the set must also be rewritten.

The HDR1 label of a rewritten file need not be rewritten. To verify the HDR1 label but not overwrite the label, specify LPROC=,R and ACS=,W on the Q5LABEL call.

For example, suppose a multifile set named FILESET has three files. The following calls in a program replace the data in the second file.

```
CALL Q5RQUEST(`LFN=`, `FILESET`, `DT=`, `NT`,
+ `VSN=`, `VSNLIST`, `ACS=`, `RW`, `RT=`, `B`),
CALL Q5LABEL(`LFN=`, `X`, `MFN=`, `FILESET`,
+ `FSN=`, 3)
CALL Q5LABEL(`LFN=`, `Y`, `MFN=`, `FILESET`,
+ `FSN=`, 2, `LPROC=`, `R`, `ACS=`, `W`)
CALL Q5LABEL(`LFN=`, `Z`, `MFN=`, `FILESET`,
+ `FSN=`, 9999)
CALL Q5OPEN(`LFN=`, `X`)
CALL Q5GETB(`LFN=`, `X`, `WSA=`, `TEMP`, `WSL=`, 512).
CALL Q5CLOSE(`LFN=`, `X`)
CALL Q5OPEN(`LFN=`, `Y`)
CALL Q5PUTB(`LFN=`, `Y`, `WSA=`, `NEW`, `WSL=`, 512).
CALL Q5CLOSE(`LFN=`, `Y`)
CALL Q5OPEN(`LFN=`, `Z`)
CALL Q5PUTB(`LFN=`, `Z`, `WSA=`, `TEMP`, `WSL=`, 512).
CALL Q5CLOSE(`LFN=`, `Z`)
```

The Q5RQUEST call specifies the multifile set name, the tape volume containing the multifile set, and read and write access for the files in the set. The first Q5LABEL call is used to read the third file in the set. The second Q5LABEL call is used to rewrite the data in the second file without rewriting the HDR1 label of the file [the label processing option (LPROC) is read, but the data access (ACS) is write]. The third Q5LABEL call is used to rewrite the third file (data and labels). It specifies file sequence number 9999 because the file is appended to the multifile set.

The first Q5OPEN call opens the third file in the set; the Q5GETB call copies data from the third file to an array named TEMP so that the data is not lost. (For the purposes of this example, assume that each file has only one block.) The second Q5OPEN call opens the second file; the first Q5PUTB call overwrites the data in the second file with the data from array NEW. The third Q5OPEN call opens the third file; the second Q5PUTB call rewrites the data saved in the TEMP array.

Q5MAPIN - MAP IN VIRTUAL SPACE

Call the Q5MAPIN routine (refer to figure 9-19) to associate a virtual address range with a mass storage file. Specify as the virtual address range an array declared within the program. Q5MAPIN associates the array with the specified mass storage file. For more information, refer to Implicit I/O in this chapter.

Read or write access to the mass storage file is required, and the file for implicit I/O must be opened.

SIL, by default, maps into the drop file data areas not mapped into other files. You can also specify the drop file on a Q5MAPIN call. Q5MAPIN cannot map in a controllee file.

SIL changes the record type in the file's FIT to undefined when it opens the file for implicit I/O. It does not change the record type in the file's file index entry.

If the region being mapped is not covered by the file and the file is nonextendable with write access permitted, a fatal error is returned (status code 1539).

Q5MAPIN cannot map in a file connected to a terminal. An attempt to do so returns a fatal error (status code 1458).

Q5MAPIN issues the MAP system message.

Call Format

```
CALL Q5MAPIN( { ^LFN=,lfn  
               ^FLUN=,flun } , ^VBA=,vba, ^LEN=,ln, optional parameters )  
               ^DROPF
```

Calling Parameters

^LFN=,lfn	Name of the mass storage file. LFN=,lfn must be specified if FLUN=,flun and DROPF are omitted.
^FLUN=,flun	Logical number SIL assigned to the file. FLUN=,flun must be specified if LFN=,lfn and DROPF are omitted.
^DROPF	Indicates that SIL should map in the drop file. DROPF must be specified if LFN=,lfn and FLUN=,flun are omitted.
^EXT=,ext	File extendability indicator. If EXT=,ext is omitted, the file is extendable. ^Y The file is extendable. ^N The file is not extendable.
^LEN=,ln	Length in 512-word blocks of the virtual region specified by the VBA=,vba parameter. LEN=,ln must be specified.

Figure 9-19. Q5MAPIN Call Format (Sheet 1 of 2)

<u>Calling Parameters</u>	
<code>^LMA=^,lma</code>	Number (relative to the beginning of the file) of the first file block to be associated with the virtual region. If <code>LMA=,lma</code> is omitted, SIL assumes <code>lma</code> is zero and uses the first sector of the file.
<code>^LPAGE^</code>	Indicates that SIL should map in 128-block units (large pages). If <code>^LPAGE^</code> is omitted, SIL maps in 1, 4, or 16-block units (small pages).
<code>^VBA=^,vba</code>	Address of the first word of the array to be mapped in as the virtual region (subscripted array name not entered as a literal). The array must be at least 512 words (one block) and it must begin on a page boundary. <code>VBA=,vba</code> must be specified.
<u>Return Parameters</u>	
<code>^CONT=^,cont</code>	Indicates whether the file is contiguous. SIL can return the following ASCII values: Y The file is contiguous. N The file is not contiguous.
<code>^ERRLEN=^,len</code>	Error message length in bytes (integer).
<code>^ERRMSG=^,msg</code>	Error message. The variable <code>msg</code> must be 80 bytes long.
<code>^EXT=^,ext</code>	File extendability indicator (ASCII, left justified, blank filled). Y The file is extendable. N The file is not extendable.
<code>^RLen=^,rlen</code>	Returns the effective length of the mapped virtual region.
<code>^STATUS=^,stat</code>	Status code. Possible values: 0 through 199, 204, 205, 303, 1421, 1458, 1516, 1519, 1537, 1538, 1548, 1551, 1553, 1560, 1707, 1708, 1813.

Figure 9-19. Q5MAPIN Call Format (Sheet 2 of 2)

Q5MAPOUT - MAP OUT VIRTUAL SPACE

Call the Q5MAPOUT routine (refer to figure 9-20) to disassociate a virtual address range from a mass storage file.

If you have write access to the mass storage file, SIL writes the modified data over the original data. If you do not have write access to the file, SIL discards the modified data at program termination.

The mass storage file mapped out can be one of your files, the task drop file, or the controllee file being executed.

After a virtual region is mapped out, references to addresses within the region no longer cause the operating system to transfer the corresponding data on mass storage into central memory. However, the program can reference the data that was implicitly read into the region while it was mapped in. Changes to the data are not written on the mass storage file.

Call Format

```
CALL Q5MAPOUT( { ^LFN=,lfn  
                ^FLUN=,flun  
                ^DROPF  
                ^CONTF } , ^VBA=,vba, ^LEN=,ln, optional parameters)
```

Calling Parameters

^LFN=,lfn	Name of the mass storage file. LFN=,lfn must be specified if FLUN=,flun, DROPF, and CONTF are omitted.
^FLUN=,flun	Logical number SIL assigned to the file. FLUN=,flun must be specified if LFN=,lfn, DROPF, and CONTF are omitted.
^DROPF	Indicates that SIL should map out the drop file. DROPF must be specified if LFN=,lfn, FLUN=,flun, and CONTF are omitted.
^CONTF	Indicates that SIL should map out the controllee file. CONTF must be specified if LFN=,lfn, FLUN=,flun, and DROPF are omitted.
^VBA=,vba	Name of the array to be mapped out. VBA=,vba must be specified.
^LEN=,ln	Length of the array in 512-word blocks. LEN=,ln must be specified.

Return Parameters

^ERRLEN=,len	Error message length in bytes (integer).
^ERRMSG=,msg	Error message. The variable msg must be 80 bytes long.
^STATUS=,stat	Status code. Possible values: 0 through 199, 204, 205, 303, 1421, 1458, 1516, 1519, 1550, 1551, 1553, 1560, 1561, 1562, 1565, 1566, 1707, 1708.

Figure 9-20. Q5MAPOUT Call Format

Q5MAPOUT cannot map out a file connected to a terminal. An attempt to do so returns a fatal error (status code 1458).

Q5MAPOUT issues the MAP system message.

Q5OPEN - OPEN FILE

Call the Q5OPEN routine (refer to figure 9-21) to open a file. Opening a file allows the task to read or write data on the file.

A file index entry must exist for the file opened. A file index entry is created when a file is requested, attached, or defined.

Opening a file requires a FIT. If a FIT exists for the file, the existing FIT is used. Otherwise, Q5OPEN generates a FIT for the file, using information from the file index entry and parameter values specified on the Q5OPEN call. Q5OPEN parameter values do not change the file index entry.

Q5OPEN issues the OPEN FILE system message.

Access Modes

The ACS parameter on the Q5OPEN call specifies the type of access allowed to the file while it is open. The access modes requested must be the same access modes or a subset of the access modes in the file index entry.

Shared Access

A privileged open call must specify whether other jobs are to be allowed to open the file concurrently. Other jobs cannot open the file for write access.

A privileged user's job can open a file for read access while another job also has the file open for read access. However, if the privileged user's job requests write or append access or if another user has write or append access to the file when the privileged user's job attempts to open the file, the request returns a fatal error (status code 1730).

Nonprivileged Call Format

CALL Q5OPEN({ ^LFN=,lfn
 ^FLUN=,flun } , optional parameters)

Privileged Call Format

CALL Q5OPEN({ ^LFN=,lfn
 ^FLUN=,flun } , ^SA=,sa, optional parameters)

Calling Parameters

^LFN=,lfn File name. LFN=,lfn must be specified if FLUN=,flun is omitted.

^FLUN=,flun Logical number SIL assigned to the file. FLUN=,flun must be specified if LFN=,lfn is omitted.

^ACS=,acs Access mode requested.

 ^R Read access

 ^W Write access

 ^RW
 or Read and write access
 ^WR

 ^A Append access

 ^RA
 or Read and append access
 ^AR

 ^M Modify access

 ^RM
 or Read and modify access
 ^MR

If ACS=,acs is omitted, Q5OPEN opens the file for the access specified in the FIT for the file, if one exists; otherwise, it opens the file for the access specified when the file was requested, attached, or defined.

^BUFL1=,bll Length of buffer 1 in 512-word blocks. If BUFL1=,bll is omitted, the installation-defined buffer length is used (the release value is 8 blocks). The values are 1 to 24*N, where N is the number of blocks per page (with N=1, 4, 16 or 128). The value is validated when I/O is performed.

Figure 9-21. Q5OPEN Call Format (Sheet 1 of 9)

Calling Parameters

^BUFL2=^,b12	Length of buffer 2 in 512-word blocks. If BUFL2=,b12 is omitted, the installation-defined buffer length is used (the release value is 8 blocks). The values are 1 to 24*N, where N is the number of blocks per page (with N=1, 4, 16 or 128). The value is validated when I/O is performed.
^BUF1=^,b1	Array to be used as data buffer 1. The buffer must be on a block boundary (specified by a LOAD utility parameter). If the buffer is mapped on a large page, it must not cross a large page boundary.
^BUF2=^,b2	Array to be used as data buffer 2. The buffer must be on a block boundary (specified by a LOAD utility parameter). If the buffer is mapped on a large page, it must not cross a large page boundary.
^MNR=^,mnr	Minimum record length in bytes. For record types other than F, SIL checks to ensure that a record is not shorter than this value. SIL does not use this value when writing F format records. If MNR=,mnr is omitted, SIL assumes that the maximum record length is zero bytes.
^MXR=^,mxr	Maximum record length in bytes. For F format records, mxr is the fixed record length. For other record types, SIL checks to ensure that the record is not longer than this value. If MXR=,mxr is omitted, SIL assumes that the maximum record length is the default set by an installation parameter.
^WAIT=^,wait	Indicates whether the task waits for a file that is currently opened by another task. Y Suspends the task to wait until the file is available. The amount of time the task is suspended is determined by installation parameters. The default is Y. N Does not wait for the file.
^WSA=^,wsa	Working storage area used by get and put I/O calls.
^WSL=^,wsl	Length (in bytes) of the working storage area.

Calling Parameters for Mass Storage Files Only

^EXT=^,ext	File extendability indicator. If EXT=,ext is omitted, the file is extendable. ^Y^ The file is extendable. ^N^ The file is not extendable.
------------	---

Figure 9-21. Q5OPEN Call Format (Sheet 2 of 9)

Calling Parameters for Mass Storage Files Only

^IMP^	Indicates that SIL should open the file for implicit I/O. The record format in the file's FIT is changed to undefined; the record format in the file's FILEI entry is not changed. If IMP is omitted, the file is opened for explicit I/O.
^MODDROP^	Indicates that modified pages of a file opened for implicit I/O are written to the task drop file rather than to the original file. If MODDROP is specified, IMP must also be specified. If MODDROP is omitted, modified pages are written to the original file.
^NOCOMP^	Indicates that SIL should not perform blank compression and expansion on this file. If NOCOMP is omitted, SIL performs blank compression and expansion.
^SFO=^,fo	File organization. If SFO=,fo is omitted, SIL assumes the installation-defined default organization (released value, sequential access).
	^D^ Direct access
	^S^ Sequential access

Calling Parameters for Mass Storage and Tape Files Only

^BT=^,bt	Blocking type. If BT=,bt is omitted, the file has character count blocking.
	^C^ Character count blocking
	^I^ Internal blocking
	^K^ Record count blocking
^OFF=^,ofp	File positioning when the file is opened.
	If the file is a direct access file, the parameter specification is ignored; the file is not rewound.
	^N^ Do not rewind the file. (Default.)
	^R^ Rewind the file.
^PC=^,pc	Padding character for F format records. If PC=,pc is omitted, SIL pads with the installation-defined character (released value, blank).
^RMK=^,rmk	Record delimiting character for R format records. If RMK=,rmk is omitted, SIL uses the installation-specified character (usually ASCII US, #1F).

Figure 9-21. Q5OPEN Call Format (Sheet 3 of 9)

Calling Parameters for Mass Storage and Tape Files Only

`^RT=^,rt` Record type. If SFO=,D is specified, the only valid record format is F. If RT=,rt is omitted, Q5OPEN uses the existing record type for the file.

- `^B^` System block (tape files only)
- `^F^` ANSI fixed length
- `^L^` CYBER Record Manager control word (tape files only)
- `^R^` Record mark delimited
- `^U^` Undefined
- `^W^` Control word

Calling Parameters for Tape Files Only

`^ADO=^,ado` Assembly/disassembly option used for CYBER 170/CYBER 200 binary tape interchange (ASCII, left justified, blank filled). If selected, each 60 bits of tape data read is stored in a CYBER 200 64-bit word with the upper 4 bits as zero.

- `^BI^` Binary; no assembly/disassembly performed.
- `^BW^` 60 to 64; assembly/disassembly performed.

If ADO=,ado is omitted, BI is used for all files except files with L record type and I blocking, for which BW is the default.

NOTE

The assembly/disassembly option (ADO=,BW) is invalid if the data conversion option is selected (CONVERT). Assembly/disassembly is valid only for binary data; it is not valid for character coded data.

`^CONVERT=^,tm` Data conversion mode; indicates whether the tape data is to be stored as binary or character data.

- `^Y^` Character data; convert data to and from character codes.
- `^N^` Binary data; do not convert data.

Figure 9-21. Q5OPEN Call Format (Sheet 4 of 9)

Calling Parameters for Tape Files Only

<code>^ETP^</code>	<p>End of tape processing option. If ETP is specified, control returns to you when the end of a tape volume is encountered. You can then call Q5REELSW to change tape volumes.</p> <p>If ETP is omitted, the system switches tape volumes automatically when it encounters the end of a tape volume.</p>
<code>^LABA=^,addr</code>	<p>Address of an array to which Q5OPEN returns the labels it writes or has read. The array must begin on a word boundary. If LABA=,addr is omitted, Q5OPEN does not return the labels written.</p>
<code>^LABL=^,n</code>	<p>Number of words in the array to which Q5OPEN returns the labels it writes. The length should be a multiple of 10; if it is not, the length is reduced to a multiple of 10. The valid range is from 10 through 510 words. To view the HDR1 label, the length must be at least 20.</p> <p>If LABL=,n is omitted, the length is assumed to be 10 words.</p>
<code>^TAPETBA=^,adr</code>	<p>Address of the array to which each tape I/O request copies the current system tapes table entry for the file. The array must begin on a word boundary. If TAPETBA=,adr is omitted, tape I/O requests do not copy the tapes table entry.</p>
<code>^TAPETBL=^,n</code>	<p>Number of words in the tapes table entry array (integer). Each tapes table entry requires twelve words.</p>
<code>^UEP^</code>	<p>User error processing option. If UEP is specified, a tape I/O request returns control to you when it detects a tape I/O error whose code is in the range 101 through 299. The error code is returned in the error message for status code 1476. Appendix B of this manual lists the meanings of the error codes.</p> <p>If UEP is omitted, a tape I/O error gives control to the system for standard error recovery.</p>
<code>^ULABA=^,addr</code>	<p>Address of an array containing user-specified labels to be written after the HDR1 label. The array must begin on a word boundary. If ULABA=,addr is omitted, Q5OPEN writes no user-specified labels.</p>
<code>^ULABL=^,n</code>	<p>Number of words in the array containing the user-specified labels. The length should be a multiple of 10; if it is not, the length is reduced to a multiple of 10. The valid range is from 10 through 510 words.</p> <p>If ULABL=,n is omitted, the length is assumed to be 10 words.</p>
<code>^VSN=^,adr</code>	<p>Address of the array to which Q5OPEN returns the VSN list for the file. The array must begin on a word boundary. If VSN=,adr is omitted, Q5OPEN does not return the VSN list.</p>

Figure 9-21. Q5OPEN Call Format (Sheet 5 of 9)

Calling Parameters for Tape Files Only

`^VSNL=^,n` Number of words in the VSN list array (integer). Each VSN requires one word. If `VSNL=,n` is omitted, the array is assumed to be one word long.

Calling Parameters for Privileged Users Only

`^FADE=^,array` Name of the 16-word array to receive a copy of the file access directory entry for the file if one exists. If a file access directory entry does not exist for the file, Q5OPEN sets the first two words of the array to binary zero.

`^FITE=^,array` Name of array to receive a copy of the file index entry for the opened file. The first word of the array must contain the owner's user number or the name of the pool to which the file belongs. The second word must contain the file name.

`^FITEL=^,alen` Length (in words) of the array named by the `FITE=,array` parameter. The system checks that this length is the length of a file index entry.

`^SA=^,sa` Shared access status. This parameter is required for privileged open calls. It cannot be specified if `IMP` is specified.

`^Y` Other tasks can open the file for other than write access.

`^N` No other task can open the file.

Return Parameters

`^DT=^,dt` Device type on which the file resides (ASCII, left justified, blank filled).

 MS Mass storage

 NT Magnetic tape

 TE Interactive terminal

`^ERRLEN=^,len` Error message length in bytes (integer).

`^ERRMSG=^,msg` Error message. The variable `msg` must be 80 bytes long.

`^RACS=^,acs` Access granted (ASCII, left justified, blank filled).

 R Read access

 W Write access

 RW
 or Read and write access
 WR

Figure 9-21. Q5OPEN Call Format (Sheet 6 of 9)

Return Parameters

A	Append access
RA or AR	Read and append access
M	Modify access
RM or MR	Read and modify access
'RFLUN=',flun	Logical number SIL assigned to the file and its FIT.
'RRT=',rrt	Record type returned (left-justified, blank filled in a word).
B	System block
F	ANSI fixed length
L	CYBER Record Manager control word
R	Record mark delimited
U	Undefined
W	Control word
'SLEV=',sl	Security level of the file (integer from 1 through 8).
'STATUS=',stat	Status code. Possible values: 0 through 199, 203, 204, 205, 252, 258, 300, 303, 1402, 1411, 1425, 1427, 1428, 1454, 1459, 1460, 1472, 1474 thru 1476, 1480 thru 1487, 1517, 1518, 1522, 1525, 1529, 1541, 1543, 1560, 1570, 1616, 1619, 1620, 1628, 1629, 1650, 1685, 1687, 1726, 1730, 1735, 1800.

Return Parameters for Mass Storage Files Only

'CONT=',con	Contiguity requirement for the file (ASCII, left-justified, blank filled).
Y	The file must be contiguous.
N	The file can be noncontiguous (segmented).
'EXT=',ext	File extendability indicator (ASCII, left-justified, blank filled).
Y	The file is extendable.
N	The file is not extendable.

Figure 9-21. Q5OPEN Call Format (Sheet 7 of 9)

Return Parameters for Mass Storage Files Only

'FC=',cat File category (ASCII, left-justified, blank filled).

 BA Batch input file

 MS Mass storage file

 OT Output file

 SD System-generated drop file

 SR Scratch file

 UD User-generated drop file

 UN Undefined file

 WT Modified drop file

'LEN=',fl Amount of space allocated to the file and known to the system. If a file is segmented and some segments are on devices that are down, the length of the segment available is returned.

'LP=',lp File duration (ASCII, left-justified, blank filled).

 L Local (job duration file)

 P Permanent file

'OCAT=',own File ownership category (ASCII, left-justified, blank filled).

 PO Pool file

 PR Private file

 PU Public file

'PN=',pn Six-character identifier of the pack on which the file resides.

'REXT=',rext File extendability indicator (ASCII, left-justified, blank filled).

 Y The file is extendable

 N The file is not extendable

'TYPE=',typ File type (ASCII, left-justified, blank filled).

 PD Physical data file

 VC Virtual code (controllee) file

Figure 9-21. Q5OPEN Call Format (Sheet 8 of 9)

Return Parameters for Tape Files Only

^RFID=`,fid	File identifier from HDR1 label (17 ASCII characters); applies to ANSI labeled tape files only.
^RFSN=`,rfsn	File sequence number from HDR1 label (integer); applies to ANSI labeled tape files only.
^RLABL=`,n	Length in words of the labels written (integer). The length returned is always a multiple of 10. If RLABL=,n is omitted, Q5OPEN does not return the length.
^RVSNL=`,n	Length in words of the VSN list returned (integer).

Figure 9-21. Q5OPEN Call Format (Sheet 9 of 9)

I/O Buffers

By specifying the BUF1=,b1 and BUF2=,b2 parameters, the virtual address range that is to be the I/O buffer for the file can be assigned. The BUF1=,b1 and/or BUF2=,b2 parameters must be specified if you expect to explicitly read or write the opened file. After the task opens the file and executes the first I/O request for the file, the buffer could contain file data at any time until the file is closed. SIL flushes the I/O buffers when the file is closed.

NOTE

Do not assign the same virtual memory space as an I/O buffer for two files that are open at the same time. Data for one file could overwrite data for the other file.

Implicit I/O

A Q5OPEN call can open a mass storage file for explicit I/O or implicit I/O. When opening a file for implicit I/O, specify read or read and write access. An attempt to open a file for implicit I/O without read access returns a fatal error (status code 1620).

A mass storage file for implicit I/O can be opened. All modified pages of the file can be specified to write to the task drop file instead of to the opened file. To do so, specify the IMP and MODDROP parameters on the Q5OPEN call. The IMP parameter must be specified in the MODDROP parameter; otherwise, Q5OPEN returns a fatal error (status code 1428).

Files Connected to Terminals

Only a level 2 controllee of the interactive processor can open a file connected to a terminal. If a controllee at a lower level in a controllee chain attempts to do so, a fatal error is returned (status code 1459).

Q5OPEN cannot open a file connected to a terminal for implicit I/O. A Q5OPEN call that specifies the TE device type and the IMP parameter returns a fatal error (status code 1460).

If a Q5OPEN call to open a file connected to a terminal specifies mass storage or tape parameters, Q5OPEN returns a warning error (status code 1454).

Tape Files

A tape file must be requested before it is opened. Q5OPEN checks to see that the specified blocking type (BT) and record type (RT) do not conflict with the tape format specified when the tape file was requested. The valid combinations of tape format, block type, and record type are shown in table 9-3. If the specified values conflict, Q5OPEN returns a fatal error (status code 1484).

For V format files, Q5OPEN checks to see that the specified buffers are at least as large as the MPRU size. If a specified buffer is too small, Q5OPEN returns a warning error (status code 1486).

When a tape file request specifies read-only access, the tape volume is mounted without a write ring. If the subsequent Q5OPEN call for the file specifies write access, Q5OPEN returns a fatal error (status code 1481).

If the Q5OPEN call specifies tape parameters for a nontape file, SIL returns a warning error (status code 1472).

By default, the system processes fatal tape I/O errors. However, if the Q5OPEN call specifies the UEP parameter, a fatal tape I/O error returns control to you. It returns the fatal error status code 1476. The status message specifies a tape I/O error code in the range 101 through 299. The tape I/O error codes and their meanings are listed in appendix B of this manual. To clear a tape error status, call the Q5CLIOER routine.

Multivolume Tape Files

A tape file can extend for more than one tape volume. The tape volumes assigned to the file are specified as a list of volume serial numbers (VSNs) when the file is requested. If the Q5OPEN call specifies an array with the VSNA and VSNL parameters, Q5OPEN returns the VSN list in the array. If the Q5OPEN call specifies the RVSNL parameter, Q5OPEN returns the number of VSNs returned in the VSN list array.

By default, the system automatically switches to the next tape volume in the VSN list when it encounters the end of a tape volume. However, if the Q5OPEN call specifies the ETP parameter, the system returns control to you when it encounters the end of a tape volume. You can then call the Q5REELSW routine to switch to another tape volume.

Tape Label Processing

For ANSI labeled tape files, the specified access mode (ACS) must not conflict with the label processing option specified when the file was requested. If the access mode is read only and labels are to be written, Q5OPEN returns a fatal error (status code 1480).

Assuming that the label processing option (LPROC) indicates that the existing labels are to be read and verified, Q5OPEN searches for an HDR1 label when it opens the file. If it does not find an HDR1 label, it returns a fatal error (status code 1482). If it finds illegal user labels, it also returns a fatal error (status code 1475).

When opening a file in a multifile set, Q5OPEN searches for an HDR1 label containing the file sequence number and file identifier specified on the file request. If it cannot find the file, it returns a fatal error (status code 1485).

When Q5OPEN opens an ANSI labeled tape file for write access and write label processing (LPROC=W), it writes an HDR1 label to indicate the beginning of the file. The content of the label is copied from the FIT, as specified when the file was requested.

Q5OPEN returns the contents of the labels it reads or writes if the Q5OPEN call specifies an array using the LABA and LABL parameters. The array should be long enough for all labels written (80 bytes for the HDR1 label plus the length of any user-specified labels). If the array is too short, SIL returns a warning error (status code 1474).

If the Q5OPEN call specifies the RLABL parameter, Q5OPEN returns the length of the labels written.

User-Specified Labels

If the Q5OPEN call specifies the ULABA parameter, Q5OPEN writes the contents of the specified array as additional header labels after the HDR1 label. The valid user-specified ANSI header labels are HDRn and UHLn. (The ANSI label formats are shown appendix F of this manual.)

If user-specified label array contains invalid labels, Q5OPEN returns a fatal error (status code 1475).

Q5PATAch - ATTACH POOL

Call the Q5PATAch routine (refer to figure 9-22) to attach an existing pool. Attaching a pool attaches all files that belong to the pool and that have a security level not greater than that of the task.

Only the pool boss or a pool member can attach a pool. (A pool member is a user granted access by the pool boss.) The type of access allowed for a pool file is defined when the file is given to the pool. For more information, refer to Pool Files in chapter 2 of this manual.

A task can have up to four pools attached at the same time (including the system pool).

Q5PATAch issues the POOL FILE MANAGER system message.

<u>Call Format</u>	
CALL Q5PATAch(`POOL=`,pool, optional parameters)	
<u>Calling Parameters</u>	
`POOL=`,pool	Name of pool to be attached. The name must be left justified and blank filled. This parameter is required.
<u>Return Parameters</u>	
`ERRLEN=`,len	Error message length in bytes (integer).
`ERRMSG=`,msg	Error message. The variable msg must be 80 bytes long.
`STATUS=`,stat	Status code. Possible values: 0 through 199, 204, 301, 1503, 1508, 1544, 1545, 1618.

Figure 9-22. Q5PATAch Call Format

Q5PCREAT - CREATE POOL

Call the Q5PCREAT routine (refer to figure 9-23) to create a pool. Q5PCREAT adds the specified name to the pool list. The user who creates the pool is the pool boss.

Initially, no files belong to the newly created pool. A file can be given to the pool with a Q5GIVE call. For more information, refer to Pool Files in chapter 2 of this manual.

Q5PCREAT issues the POOL FILE MANAGER system message.

Call Format

CALL Q5PCREAT('POOL=',pool, optional parameters)

Calling Parameters

'POOL=',pool Name of the new pool. The name must be eight letters and digits, starting with a letter, left justified and blank filled. This parameter is required.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).
'ERRMSG=',msg Error message. The variable msg must be 80 bytes.
'STATUS=',stat Status code. Possible values: 0 through 199, 204, 1507, 1508, 1511.

Figure 9-23. Q5PCREAT Call Format

Q5PDESTR - DESTROY POOL

Call the Q5PDESTR routine (refer to figure 9-24) to remove the specified pool name from the pool list.

Only the pool boss can remove a pool name; no files can belong to the pool, and the pool cannot be attached to a task (including the task issuing the Q5PDESTR call).

Q5PDESTR issues the POOL FILE MANAGER system message.

<u>Call Format</u>	
CALL Q5PDESTR(^POOL=^,pool, optional parameters)	
<u>Calling Parameters</u>	
^POOL=^,pool	Name of the new pool. The name must be left justified and blank filled. This parameter is required.
<u>Return Parameters</u>	
^ERRLEN=^,len	Error message length in bytes (integer).
^ERRMSG=^,msg	Error message. The variable msg must be 80 bytes long.
^STATUS=^,stat	Status code. Possible values: 0 through 199, 204, 301, 1508, 1531, 1547.

Figure 9-24. Q5PDESTR Call Format

Q5PDTACH - DETACH POOL

Call the Q5PDTACH routine (refer to figure 9-25) to return an attached pool of files. All files in the pool need not be closed before the job returns the pool.

Q5PDTACH issues the POOL FILE MANAGER system message.

<u>Call Format</u>	
CALL Q5PDTACH(^POOL=^,pool,optional parameters)	
<u>Calling Parameters</u>	
^POOL=^,pool	Pool name, left justified and blank filled. This parameter is required.
<u>Return Parameters</u>	
^ERRLEN=^,len	Error message length in bytes (integer).
^ERRMSG=^,msg	Error message. The variable msg must be 80 bytes long.
^STATUS=^,stat	Status code. Possible values: 0 through 199, 204, 301, 1506.

Figure 9-25. Q5PDTACH Call Format

Q5PERMIT - CHANGE ACCESS PERMISSION SET

Call the Q5PERMIT routine (refer to figure 9-26) to change an access permission set.

Only the file owner can change the access permission sets of a private file. Only the pool boss can change the access permission sets of a file belonging to a pool. Only a privileged user can change the access permission set of a public file.

To change an access permission set of a private file, the file must be attached. The pool boss need not attach the pool to change the access permission set of a pool file.

The parameters PRODTN= and OWNER= allow the site security administrator user number to grant or remove production status for a file. These parameters are mutually exclusive with the access permission parameters ACS= and OWNER= . Refer to chapter 7 of the Installation Handbook for details.

Tape Access

Assuming that a tape file request specifies read and write access to the file, a Q5PERMIT call can change the requested access mode for the file to read only or write only. For example, if read and write access is requested for a multifile set, Q5PERMIT calls could change the access mode to read only or write only for each file in the set.

Append, modify, and execute access are not valid for a tape file. If a Q5PERMIT call specifies A, M or X access, Q5PERMIT returns a fatal error (status code 1630).

If a Q5PERMIT call specifies an access mode not specified when the file was requested, Q5PERMIT returns a fatal error (status code 1631).

Call Format

```
CALL Q5PERMIT( { ^LFN=,lfn, ^ACS=,acs,optional parameters }  
               { ^FLUN=,rflun, }
```

Calling Parameters

^LFN=,lfn	Name of the file. LFN=,lfn must be specified if FLUN=,rflun is omitted.
^FLUN=,rflun	Number SIL assigned to the file. FLUN=,rflun must be specified if LFN=,lfn is omitted.
^ACS=,acs	New access permission set. To grant no access to the file, specify ^ACS=,^NONE. To grant one or more access permissions, acs must be a string composed of the following letters without separators: R Read permission W Write permission X Execute permission A Append permission M Modify permission

Figure 9-26. Q5PERMIT Call Format (Sheet 1 of 2)

Calling Parameters

`^POOL=^,poolname` Name of the pool to which the file belongs (ASCII characters, left justified, blank filled). POOL=,poolname and PUBLIC are mutually exclusive. If POOL=,poolname and PUBLIC are omitted, the file must be a private file.

`^PUBLIC^` Indicates that the file is a public file. POOL=,poolname and PUBLIC are mutually exclusive. If POOL=,poolname and PUBLIC are omitted, the file must be a private file.

`^USER=^,user` Indicates the access permission sets changed.

`userno` Individual access permission set for the specified user number (ASCII, left justified, blank filled). This option is valid only for a private permanent file.

`^GENERAL^` General access permission set.

`^*^` All access permission sets.

If USER=user is omitted, the access permission set changed depends on the file category as follows:

Private Owner's access permission set.

Pool Pool boss's access permission set.

Public General access permission set.

`^PRODTN=^,prod` Only the site security administrator user number can use this parameter. Specify:

`prod=Y` To grant production status to the file; all write access permissions are removed.

`prod=N` To remove production status from the file. Access permissions are not changed.

PRODTN= is mutually exclusive with ACS= .

`^OWNER=^,ownum` Owner's ASCII user number, left-justified, blank-filled, for a private permanent file when PRODTN is specified. If omitted, the site security administrator is assumed to be the owner.

Return Parameters

`^ERRLEN=^,len` Error message length in bytes (integer).

`^ERRMSG=^,msg` Error message. The variable msg must be 80 bytes.

`^STATUS=^,stat` Status code. SIL returns one of the following integers: 0 through 199, 200, 204, 205, 210, 264, 301, 303, 306, 307, 310, 517, 518, 1402, 1425, 1546, 1630, 1631, 1682, 1691, 1727, 1728, 1731.

Figure 9-26. Q5PERMIT Call Format (Sheet 2 of 2)

Q5PGRACC - GRANT ACCESS TO POOL

Call the Q5PGRACC routine (refer to figure 9-27) to grant other users access to a pool. Only the pool boss for the specified pool can grant users access.

Q5PGRACC can grant access to all users or to a specified list of users. The type of file access allowed is defined when the file is given to the pool.

Q5PGRACC issues the POOL FILE MANAGER system message.

<u>Call Format</u>	
CALL Q5PGRACC(^POOL=^,pool, optional parameters)	
<u>Calling Parameters</u>	
^POOL=^,pool	Pool name, left justified and blank filled. This parameter is required.
^NU=^,nu	Number of user numbers in the list specified by the ULIST=,ul parameter. If NU=,nu is omitted, SIL assumes that the ULIST=,ul parameter does not specify a list of user numbers.
^ULIST=^,ul	List of user numbers to be granted access to the pool. The numbers must be in integer format, one user number per word. If ULIST=,ul is omitted, SIL grants all user numbers access to the pool.
<u>Return Parameters</u>	
^ERRLEN=^,len	Error message length in bytes (integer).
^ERRMSG=^,msg	Error message. The variable msg must be 80 bytes long.
^STATUS=^,stat	Status code. Possible values: 0 through 199, 204, 301, 1508, 1547.

Figure 9-27. Q5PGRACC Call Format

Q5POOLS - LIST POOLS

The entry format for the Q5POOLS routine is defined in figure 9-28. Call the Q5POOLS routine (refer to figure 9-29) to obtain a list of all pools and pool bosses. Q5POOLS copies the list into the specified buffer. It copies a two-word entry for each pool.

Q5POOLS can return the number of words copied to the buffer. Divide this number by two to obtain the number of pool entries copied.

Q5POOLS issues the POOL FILE MANAGER system message.

pcount	16	res	16	pfree	12	pboss	20
poolname							64

pcount	Number of users that have the pool attached
res	Reserved
pfree	Currently unused
pboss	User number of pool boss (binary format)
poolname	Pool name (eight ASCII characters)

Figure 9-28. Pool List Entry Format

<u>Call Format</u>	
CALL Q5POOLS(^BUFFER= ^,bfr, ^BUFLEN= ^,bl, optional parameters)	
<u>Calling Parameters</u>	
^BUFFER= ^,bfr	Array to which the pool and pool boss names are copied. This parameter is required.
^BUFLEN= ^,bl	Length of the array specified by the BUFFER=,bfr parameter. This parameter is required.
<u>Return Parameters</u>	
^RBUFLEN= ^,rbl	Number of words copied to the buffer.
^ERRLEN= ^,len	Error message length in bytes (integer).
^ERRMSG= ^,msg	Error message. The variable msg must be 80 bytes long.
^STATUS= ^,stat	Status code. Possible values: 0 through 199, 204, 1549.

Figure 9-29. Q5POOLS Call Format

Q5PREACC - REMOVE ACCESS TO POOL

Call the Q5PREACC routine (refer to figure 9-30) to remove pool access granted to one or more users. The user calling Q5PREACC must be the pool boss for the specified pool.

If you attach the pool when SIL processes a Q5PREACC call to remove your access, the removal of the pool access goes into effect after you have detached the pool.

Q5PREACC issues the POOL FILE MANAGER system message.

<u>Call Format</u>	
CALL Q5PREACC(´POOL=´,pool,´ULIST=´,ul,optional parameters)	
<u>Calling Parameters</u>	
´POOL=´,pool	Name of the pool from which SIL should remove access privileges. The pool name must be left justified and blank filled in the name. This parameter is required.
´ULIST=´,ul	List of user numbers whose access privileges SIL should remove. The user numbers must be in binary (right justified, zero filled). This parameter is required.
´NU=´,nu	Number of user numbers specified by ULIST=,ul. If NU=,nu is omitted, SIL assumes that ULIST=,ul specifies one user number.
<u>Return Parameters</u>	
´ERRLEN=´,len	Error message length in bytes (integer).
´ERRMSG=´,msg	Error message. The variable msg must be 80 bytes long.
´STATUS=´,stat	Status code. Possible values: 0 through 199, 204, 301, 1503, 1508, 1547.

Figure 9-30. Q5PREACC Call Format

Q5PURGE - PURGE FILE

Call the Q5PURGE routine (refer to figure 9-31) to purge a permanent file. Q5PURGE deletes the file index entry for the specified file and releases its mass storage space.

SIL assumes that the file is the private file with the specified name unless the OCAT parameter on the Q5PURGE call specifies that the file is a pool or public file.

Only a file owner or a privileged user can purge a private file. Only a privileged user or the pool boss can purge a pool file (the pool must be attached to the task). Only a privileged user can purge a public file.

A privileged user can purge other users' private and pool files.

The file must be closed before it is purged. If the purged file is attached to a job, it becomes a local file.

Q5PURGE cannot purge a file connected to a terminal. An attempt to do so returns a fatal error (status code 1463).

Q5PURGE issues the DESTROY FILE system message.

Call Format

Nonprivileged call

```
CALL Q5PURGE( { ^LFN=,lfn  
               ^FLUN=,rflun } ,optional parameters)
```

Privileged Call

```
CALL Q5PURGE( { ^LFN=,lfn  
               ^FLUN=,rflun } ,^OWNER=,un, optional parameters)
```

Calling Parameters

^LFN=,lfn	File name. LFN=,lfn must be specified if FLUN=,flun is omitted.
^FLUN=,flun	Logical number SIL assigned to the file. FLUN=,flun must be specified if LFN=,lfn is omitted.
^OCAT=,oc	File ownership category. If OCAT=,oc is omitted, SIL purges a private file.
	^PO Pool file
	^PR Private file
	^PV Public file
^POOL=,poolname	Name of the attached pool from which the file is purged. This parameter is required if OCAT=,PO is specified.

Figure 9-31. Q5PURGE Call Format (Sheet 1 of 2)

Calling Parameter for Privileged Users Only

OWNER=,own User number or pool name to which the file belongs. A user number must be an integer, right justified and zero filled; a pool name must be a character string, left justified and blank filled.

Return Parameters

ERRLEN=,len Error message length in bytes (integer).

ERRMSG=,msg Error message. The variable msg must be 80 bytes long.

STATUS=,stat Status code. Possible values: 0 through 199, 204, 205, 303, 1402, 1424, 1546, 1685, 1687, 1690, 1735.

Figure 9-31. Q5PURGE Call Format (Sheet 2 of 2)

Q5PUSERL - LIST USERS WITH ACCESS TO POOL

Call the Q5PUSERL routine (refer to figure 9-32) to obtain a list of users having access to a pool. Only the pool boss of the specified pool can obtain the list.

Q5PUSERL copies the user numbers to the specified buffer, one user number per word in integer format.

Q5PUSERL issues the POOL FILE MANAGER system message.

<u>Call Format</u>	
CALL Q5PUSERL(´POOL=´,pool,´ULIST=´,ul,´NU=´,nu,optional parameters)	
<u>Calling Parameters</u>	
´POOL=´,pool	Name of pool whose access list SIL is to supply. The name must be left justified and blank filled. This parameter is required.
´NU=´,nu	Number of words in the buffer specified by the ULIST=,ul parameter. This parameter is required.
´RNU=´,rnu	Number of user numbers copied to the buffer specified by ´ULIST=´,ul. If all users can access the pool, Q5PUSERL returns the value 1 in the variable.
<u>Return Parameters</u>	
´ERRLEN=´,len	Error message length in bytes (integer).
´ERRMSG=´,msg	Error message. The variable msg must be 80 bytes long.
´STATUS=´,stat	Status code. Possible values: 0 through 199, 204, 301, 1508.
´ULIST=´,ul	Buffer in which Q5PUSERL returns the list of user numbers that can access the pool, one user number per word. This parameter is required. If all users can access the pool, Q5PUSERL returns one word to the buffer; the word has the value 1000000 (decimal).

Figure 9-32. Q5PUSERL Call Format

Q5PUTB - PUT A BUFFER RECORD

A Q5PUTB call (refer to figure 9-33) writes the next record on a file. If the tape file writes B-record type, Q5PUTB copies the tape block directly from the specified working storage area to the file; it does not use intermediate I/O buffers.

If the specified file is not a tape file with the B record type, a Q5PUTB call is equivalent to a Q5PUTN call.

<u>Call Format</u>	
CALL Q5PUTB({ $\begin{matrix} \text{`LFN=`,lfn} \\ \text{`FLUN=`,flun} \end{matrix} \right\}$, optional parameters)
<u>Calling Parameters</u>	
`LFN=`,lfn	File name. LFN=,lfn must be specified if FLUN=,flun is omitted.
`FLUN=`,flun	File logical unit number SIL assigned to the FIT. FLUN=,flun must be specified if LFN=,lfn is omitted.
`WSA=`,wsa	Address of the array from which Q5PUTB copies data. If WSA=,wsa is omitted, the working storage area specified in the FIT is used.
`WSL=`,wsl	Length (in bytes) of the working storage area. If WSL=,wsl is omitted, the length specified in the FIT is used.
<u>Return Parameters</u>	
`ERRLEN=`,len	Error message length in bytes (integer).
`ERRMSG=`,msg	Error message. The variable msg must be 80 bytes long.
`RSN=`,rsn	Request serial number. If RSN=,rsn is specified, Q5PUTB returns control to you immediately. The task specifies the returned RSN on a Q5CHECKB call to determine whether the Q5PUTB request has completed. If RSN=,rsn is omitted, the system suspends the task until the Q5PUTB request has completed.
`STATUS=`,stat	Status code. Possible values: 0 through 199.

Figure 9-33. Q5PUTB Call Format

If the call specifies the RSN=,rsn parameter, Q5PUTB returns control to you immediately. The task can then perform other processing while the tape I/O request completes. To determine whether the I/O request is complete, the program must call the Q5CHECKB routine.

Q5PUTN - WRITE PARTITION

Call the Q5PUTN routine (refer to figure 9-34) to transfer data from the working storage area to a physical I/O buffer. Q5PUTN appends a partition delimiter to the data.

The Q5OPEN call for the file determines the buffer used. SIL automatically writes the data to the file when the buffer is full.

If a working storage area is not specified on the Q5PUTN call, SIL uses the working storage area specified in the file's FIT. Q5PUTN transfers the amount of data specified as the working storage area length.

The file must be open for write, modify, or append access. If it is not, Q5PUTN returns a fatal error (status code 1726).

Call Format

```
CALL Q5PUTN( { ^LFN=,lfn  
              ^FLUN=,flun } ,optional parameters)
```

Calling Parameters

^LFN=,lfn	File name. LFN=,lfn must be specified if FLUN=,flun is omitted.
^PART=,part	Partition delimiter appended to the data transferred. If PART=,part is omitted, a record delimiter is appended. R Record G Group (R, W, and L formats only) F File
^FLUN=,flun	Logical number SIL assigned to the file. FLUN=,flun must be specified if LFN=,lfn is omitted.
^REC=,n	Record number (integer greater than zero). Ignored for a sequential access file. If REC=,n is omitted for a direct access file, the value in the RC FIT field plus 1 is used.
^WSA=,wsa	Working storage area. Specifying this parameter overwrites the corresponding FIT field with the specified value. If WSA=,wsa is omitted, SIL uses the working storage area specified in the file's FIT.
^WSL=,wsl	Number of bytes of data transferred (working storage area length). Specifying this parameter overwrites the corresponding FIT field with the specified value. If WSL=,wsl is omitted, SIL uses the working storage length specified in the file's FIT.

Return Parameters

^ERRLEN=,len	Error message length in bytes (integer).
^ERRMSG=,msg	Error message. The variable msg must be 80 bytes long.
^STATUS=,stat	Status code. Possible values: 0 through 199, 253, 303, 1401, 1405, 1430, 1432, 1434, 1435, 1439, 1443, 1444, 1452, 1521, 1719, 1726.

Figure 9-34. Q5PUTN Call Format

For F format records, Q5PUTN transfers the number of bytes specified in the working storage area length (WSL) field of the FIT. If WSL is less than the fixed record length (MXR) field of the FIT, Q5PUTN returns a warning error (status code 1452) and pads the record to the MXR length. If WSL is greater than MXR, Q5PUTN returns a fatal error (status code 1435) and transfers no data.

For record formats other than F, Q5PUTN checks to ensure that WSL is within the minimum and maximum lengths specified by the MNR and MXR fields of the fit before transferring the record data. If WSL is outside the specified range, Q5PUTN returns a fatal error (status code 1435).

For R format records, Q5PUTN compresses all strings comprising three or more blank characters unless the NOCOMP parameter was specified on the Q5OPEN call. For more information, refer to Record Mark Delimited (R) Record Format in chapter 2 of this manual.

For a direct access file, SIL determines the beginning of the requested record as described under File Organization in chapter 2. If writing the record would write data past the current end of the file, Q5PUTN returns a fatal error (status code 1434).

To append a record to a mass storage file, the file must be positioned after its data but before the file delimiter if there is one. To position the file, read or skip to the end of the file and then skip backward over the file delimiter. An attempt to append a record at another file position returns a fatal error (status code 1521). For more information, refer to Appending Data in this chapter.

When a mass storage file is opened for modify access, it must be positioned at the beginning of an existing record. An attempt to write a record beyond the end of the file returns a fatal error (status code 1719).

Tape Files

In general, writing a tape record is the same as writing a mass storage record. However, besides the F, R, U, and W record types, tape files can also use the B and L record types.

When writing L records, Q5PUTN adds block control word and record control words. The L record format is described in chapter 2.

A B record is an LRU; therefore, a Q5PUTN call to write a B record writes an LRU. If the tape is written using I, SI, or LB tape format, an LRU terminator is appended to the data.

Q5PUTP - WRITE PARTIAL PARTITION

Call the Q5PUTP routine (refer to figure 9-35) to transfer data from the working storage area to a physical I/O buffer. The Q5OPEN call for the file determines the buffer used. SIL automatically writes the data to the file when the buffer is full.

Call Q5PUTP to write partial records or undefined records (U format). Q5PUTP does not add a partition delimiter to the data unless the TERM parameter is specified. A partition delimiter can also be added by issuing a Q5ENDPAR call.

The file must be open for write, modify, or append access. If it is not, Q5PUTP returns a fatal error (status code 1726).

Call Format

```
CALL Q5PUTP( { 'LFN=',lfn } , optional parameters )
              { 'FLUN=',flun }
```

Calling Parameters

'LFN=',lfn	File name. LFN=,lfn must be specified if FLUN=,flun is omitted.
'FLUN=',flun	Logical number SIL assigned to the file. FLUN=,flun must be specified if LFN=,lfn is omitted.
'PART=',part	part is the partition delimiter appended to the data transferred. If PART=,part is omitted, a record delimiter is appended if TERM is specified. R Record G Group (R, W, and L formats only) F File
'REC=',n	Record number (integer greater than zero). Ignored for a sequential access file. If REC=,n is omitted for a direct access file, the value in the RC FIT field plus 1 is used.
'TERM'	Indicates that the call is the last call for the record. For R and W formats, SIL appends a partition delimiter to the data transferred. If TERM is omitted, a partition delimiter is not appended.
'WSA=',wsa	Working storage area. Specifying this parameter overwrites the corresponding FIT field with the specified value. If WSA=,wsa is omitted, SIL uses the working storage area specified in the file's FIT.
'WSL=',wsl	Number of bytes of data transferred (working storage area length). Specifying this parameter overwrites the corresponding FIT field with the specified value. If WSL=,wsl is omitted, SIL uses the working storage length specified in the file's FIT.

Figure 9-35. Q5PUTP Call Format (Sheet 1 of 2)

<u>Return Parameters</u>	
<code>^ERRLEN=^,len</code>	Error message length in bytes (integer).
<code>^ERRMSG=^,msg</code>	Error message. The variable msg must be 80 bytes long.
<code>^STATUS=^,stat</code>	Status code. Possible values: 0 through 199, 253, 303, 1401, 1405, 1430, 1432, 1434, 1435, 1439, 1443, 1444, 1451, 1452, 1461, 1521, 1719, 1726.

Figure 9-35. Q5PUTP Call Format (Sheet 2 of 2)

For F format records, Q5PUTP compares the WSL value with the fixed record length (MXR) field of the FIT. If the values do not match, it takes the following action:

<u>If</u>	<u>Then Q5PUTP</u>
WSL is less than MXR and the TERM parameter is not specified,	Writes the data and returns normal status.
WSL added to the length of the data already written to the record is less than MXR and the TERM parameter is specified,	Returns a warning error (status code 1452) and pads the record to the MXR length.
WSL is greater than MXR and the call is the first call for the record,	Returns a fatal error (status code 1435) and transfers no data.
The call is not the first call for the record and WSL added to the length of the data already written to the record, is greater than MXR,	Returns a warning error (status code 1451) and discards the excess data.

For R format files, Q5PUTP compresses all strings comprising three or more blank characters unless you specify the NOCOMP parameter on the Q5OPEN call. For more information, refer to Record Mark Delimited (R) Record Format in chapter 2 of this manual.

For a direct access file, SIL determines the beginning of the requested record as described under File Organization in chapter 2. If writing the record would write data past the current end of the file, Q5PUTN returns a fatal error (status code 1434).

When a mass storage file is opened for append access, it must be positioned after the existing file data but before the file delimiter, if there is one, before a write operation is requested. To position the file, read or skip to the end of the file and then skip backward over the file delimiter. An attempt to append a partial record at another file position returns a fatal error (status code 1521).

When a mass storage file is opened for modify access, it must be positioned at the beginning of an existing record. An attempt to write a record beyond the end of the file returns a fatal error (status code 1719).

Q5PUTP cannot write output to a file connected to a terminal. An attempt to do so returns a fatal error (status code 1461).

Tape Files

In general, writing a tape record is the same as writing a mass storage record. However, besides the F, R, U, and W record types, tape files can also use the B and L record types.

When writing L records, block control words and record control words are added to the data written on the file. The L record format is described in chapter 2.

A B record is an LRU; therefore, a Q5PUTP call with TERM specified that writes a B record writes an LRU. If the tape is written using I, SI, or LB tape format, an LRU terminator is appended to the data.

Q5READ - READ BLOCK

Call the Q5READ routine (refer to figure 9-36) to transfer one or more blocks of data from a file to a program buffer. Attach the file and open it for explicit I/O before issuing the Q5READ call. Specify the buffer to be used on the Q5READ call; otherwise, SIL uses buffer 1 as specified in the FIT.

Call Format

```
CALL Q5READ( { ^LFN=,lfn  
              ^FLUN=,flun } ,optional parameters)
```

Calling Parameters

<code>^LFN=,lfn</code>	File name. LFN=,lfn must be specified if FLUN=,flun is omitted.
<code>^FLUN=,flun</code>	Logical number SIL assigned to the file. FLUN=,flun must be specified if LFN=,lfn is omitted.
<code>^BUFFER=,bfr</code>	Array to be used as the data buffer. The buffer must be on a block boundary (specified by a LOAD utility parameter). If the buffer is mapped on a large page, it must not cross a large page boundary. If BUFFER=,bfr is omitted and BUF2 is specified, SIL uses buffer number 2; if BUFFER=,bfr and BUF2 are omitted, SIL uses buffer number 1, as specified in the FIT. If BUFFER=,bfr is specified, either BUFLN=,bfl or BYTCNT=,bytcnt must be specified.
<code>^BUFLN=,bfl</code>	Length of the buffer specified by the BUFFER=,bfr parameter in 512-word blocks. If this parameter is specified, the BYTCNT=,bytcnt parameter must be omitted. The value is 1 to 24 * N, where N is the number of blocks per page, large or small. The value is validated before the system performs READ.
<code>^BUF1</code>	Indicates that SIL should use buffer number 1, as specified in the FIT. If BUF1 is omitted and either BUFFER=,bfr or BUF2 is specified, SIL uses the specified buffer. If BUFFER=,bfr, BUF1, and BUF2 are omitted, SIL uses buffer number 1, as specified in the FIT.
<code>^BUF2</code>	Indicates that SIL should use buffer number 2, as specified in the FIT. If BUF2 is omitted and BUFFER=,bfr is specified, SIL uses the specified buffer. If BUF2 and BUFFER=,bfr are omitted, SIL uses buffer number 1, as specified in the FIT.

Figure 9-36. Q5READ Call Format (Sheet 1 of 2)

Calling Parameters

`^WAIT^` Indicates that SIL should wait for completion of this read request before returning control to the caller. If WAIT is omitted, SIL returns control immediately to the caller.

Calling Parameters for Tape Files Only

`^BYTCNT=^,bytcnt` Length of the buffer specified by the BUFFER=,bfr parameter in bytes. If this parameter is specified, the BUFLN=,bfl parameter must be omitted.

`^LRUA=^,adr` Address of an array in which Q5READ returns descriptions of the LRUs read. If LRUA=,addr is omitted, Q5READ does not return LRU descriptions.

`^LRUL=^,n` Number of words in the LRU description array; the maximum is 40 words. If SKIP is specified, it also specifies the number of LRUs to read.

`^SKIP^` If SKIP is specified, Q5READ attempts to read the number of LRUs specified by the LRUL parameter and leaves the file positioned at the beginning of the next LRU. If SKIP is omitted, Q5READ attempts to read the maximum number of full PRUs that fit in the I/O buffer and leaves the file positioned at the beginning of the next PRU.

Return Parameters

`^ERRLEN=^,len` Error message length in bytes (integer).

`^ERRMSG=^,msg` Error message. The variable msg must be 80 bytes long.

`^LEN=^,rl` Number of bytes transferred. If the operation is not complete, the value is undefined.

`^RSN=^,rsn` Number assigned to the request. A Q5CHECK call uses this identifier.

`^STATUS=^,stat` Status code. Possible values: 0 through 199, 203, 204, 206, 207, 303, 1400, 1401, 1405, 1409, 1411, 1416, 1417, 1422, 1430, 1434, 1462, 1472, 1476, 1488, 1489, 1494, 1589, 1596, 1598, 1604, 1605, 1653, 1708, 1726.

Return Parameters for Tape Files Only

`^RLRUL=^,n` If LRUA is specified, the number of entries returned in the LRU description array. If SKIP is specified, the number of LRUs read (a partial LRU is included in the count). If RLRUL=,n is omitted, the entry count or LRU count is not returned.

`^RLEVEL=^,lev` Level number of the first LRU read if Q5READ returns the LRU terminator (one ASCII character, 0 through 9 or A through F, right justified and zero filled). This parameter does not apply to V tape format.

Figure 9-36. Q5READ Call Format (Sheet 2 of 2)

Unless the WAIT parameter is specified, SIL returns control to the caller immediately after it issues the request (before the data transfer is complete). Check for completion of the data transfer with a Q5CHECK call; however, if the WAIT parameter is specified, SIL does not return control to the caller until after the data transfer is complete.

If a buffer is specified on a Q5READ call using the BUFFER= parameter, the buffers specified in the FIT are no longer defined. Subsequent Q5READ and Q5WRITE calls must specify the BUFFER= parameter.

SIL does not check to see whether program I/O buffers overlap. For instance, one buffer could extend from address 1 to address 1024 and another buffer extend from address 512 to address 1536. In this case, a read to the second buffer after a read to the first buffer would overwrite the last 512 words of data read into the first buffer.

The file must be opened for read access. If it is not, Q5READ returns a fatal error (status code 1726).

Q5READ cannot read input from a file connected to a terminal. An attempt to do so returns a fatal error (status code 1462).

For mass storage files and files connected to a terminal, Q5READ calls the EXPLICIT I/O system message. For tape files, Q5READ issues the TAPE FUNCTION system message.

Reading Tape Data

A Q5READ call for a tape file copies data from the tape to an I/O buffer. The I/O buffer cannot be more than 48 pages long. If the call specifies a longer buffer, Q5READ returns a fatal error (status code 1488).

If a Q5READ call specifies tape parameters for a nontape file, Q5READ returns a warning error (status code 1472).

Assuming that the tape is written in I, SI, or LB format, Q5READ can return the level number from the LRU terminator. It returns the level number in the variable specified on the RLEVEL parameter.

If the Q5OPEN call for the file specified user error processing (UEP), a fatal tape I/O error returns control to you. The error code is returned in the message for status code 1476.

The SKIP parameter determines whether the Q5READ call reads one or more PRUs or one or more LRUs.

Reading PRUs

If a Q5READ call omits the SKIP parameter, Q5READ reads one or more PRUs. Before reading any data, Q5READ checks to see that the specified buffer is at least as long as the MPRU size. If it is not, Q5READ copies no data to the I/O buffer and returns a fatal error (status code 1494).

Assuming that the buffer is large enough for at least one PRU, Q5READ copies a PRU. It then checks to see whether the remaining buffer space is enough for another PRU and, if it is, copies the next PRU. Q5READ continues to copy PRUs until the space remaining in the buffer is less than the MPRU size. It has then completed reading data and returns the number of bytes read in the variable specified on the LEN=len parameter, if one is specified.

After the read, the file is left positioned at the beginning of the next PRU. The next Q5READ call begins copying data at that PRU.

Reading LRUs

If a Q5READ call specifies the SKIP parameter, Q5READ reads one or more LRUs. The LRUL parameter on the call specifies the number of LRUs to read. (If the LRUL parameter is omitted, only one LRU is read.)

Q5READ then begins copying data to the I/O buffer. It continues copying data until it has either copied the number of LRUs specified by the LRUL parameter or filled the I/O buffer with data. It returns the number of LRUs read in the variable specified on the RLRUL parameter, if one is specified. It includes a partial LRU in the count. For example, if it fills the buffer by reading three complete LRUs and a partial LRU, the LRU count returned is 4.

If the last LRU read is only a partial LRU, Q5READ sets the excess data flag in the LRU description.

After copying data, Q5READ positions the file at the beginning of the next LRU. The next Q5READ call begins reading data at the beginning of the next LRU.

LRU Description Array

If the Q5READ call specifies the LRUA=,adr parameter, Q5READ copies an LRU description to the array for each complete or partial LRU read. If the LRU description is for a complete LRU, the end of LRU flag is set and the LRU level number is returned in the LRU description.

The format of an LRU description is shown in figure 9-4. The RLRUL parameter returns the number of entries returned in the array.

Writing Additional Tape Volume Labels

By default, when Q5READ encounters the end of a tape volume, the system automatically switches to the next volume in the VSN list for the file and continues reading. However, to write additional end-of-volume and beginning-of-volume labels on the file, the read must stop when it encounters the end of a volume.

To return control to the program when the end of a tape volume is encountered, specify the end of tape option (ETP) on the Q5OPEN call for the file. Q5READ returns a fatal error (status code 1489) when it encounters the end of a tape volume. A Q5REELSW call can then write additional volume labels at the end of the current volume and the beginning of the next volume. To continue reading data from the next volume, the program must call Q5READ again.

Q5REDUCE - REDUCE FILE SPACE

Call the Q5REDUCE routine (refer to figure 9-37) to reduce the length of a file to the length of the data in the file. The Q5REDUCE releases mass storage space allocated to the file which has addresses higher than the highest address accessed in the file.

Only the file owner or a privileged user can reduce the length of a file.

The length of a controllee file cannot be less than two small pages (the minus page and the register file page). An attempt to reduce a controllee file to less than two pages returns a fatal error (status code 1423).

If the file specified on a Q5REDUCE call is a file connected to a terminal, control is returned to the caller with no action taken.

Q5REDUCE issues the REDUCE FILE LENGTH system message.

<u>Call Format</u>	
CALL Q5REDUCE({ $\hat{\text{LFN}}=\text{'},\text{lfn}$ $\hat{\text{FLUN}}=\text{'},\text{flun}$ }, optional parameters)
<u>Calling Parameters</u>	
$\hat{\text{LFN}}=\text{'},\text{lfn}$	File name. LFN=,lfn must be specified if FLUN=,flun is omitted.
$\hat{\text{FLUN}}=\text{'},\text{flun}$	Logical number SIL assigned to the file. FLUN=,flun must be specified if LFN=,lfn is omitted.
$\hat{\text{NEWLEN}}=\text{'},\text{nfl}$	New file length in 512-word blocks. If NEWLEN=,nfl is omitted, the file reduces to the word with the highest address accessed.
<u>Return Parameters</u>	
$\hat{\text{ERRLEN}}=\text{'},\text{len}$	Error message length in bytes (integer).
$\hat{\text{ERRMSG}}=\text{'},\text{msg}$	Error message. The variable msg must be 80 bytes long.
$\hat{\text{LEN}}=\text{'},\text{len}$	The new length of the file in 512-word blocks.
$\hat{\text{STATUS}}=\text{'},\text{stat}$	Status code. Possible values: 0 through 199, 204, 205, 303, 1423, 1424, 1425, 1679.

Figure 9-37. Q5REDUCE Call Format

Q5REELSW - WRITE ADDITIONAL TAPE VOLUME LABELS

A Q5REELSW call (refer to figure 9-38) continues file processing with the next tape volume in the VSN list for the file.

NOTE

A Q5REELSW call is valid only after VSOS returns control to the caller after it encounters the end-of-the tape volume. It returns control only if end of tape processing (the ETP parameter) is specified on the Q5OPEN call that opened the file.

Call Format

CALL Q5REELSW({ $\begin{matrix} \text{`LFN='}, \text{lfn} \\ \text{`FLUN='}, \text{flun} \end{matrix} \right\}$, optional parameters)

Calling Parameters

$\text{`LFN='}, \text{lfn}$	File name of open tape file. If LFN=,lfn is omitted, FLUN=,flun must be specified.
$\text{`FLUN='}, \text{flun}$	File logical unit number returned when the call opened the file. If FLUN=,flun is omitted, LFN=,lfn must be specified.
$\text{`BVLABA='}, \text{addr}$	Address of the array in which Q5REELSW copies the beginning-of-volume labels written. The array must be on a word boundary. If BVLABA=,addr is omitted, Q5REELSW does not copy the beginning of volume labels.
$\text{`BVLABL='}, \text{n}$	Length in words of the array specified on the BVLABA=,addr parameter. If BVLABL=,n is omitted, the array is assumed to be 10 words.
$\text{`EVLABA='}, \text{addr}$	Address of the array in which Q5REELSW copies the end-of-volume labels written. The array must be on a word boundary. If EVLABA=,addr is omitted, Q5REELSW does not copy the end-of-volume labels.
$\text{`EVLABL='}, \text{n}$	Length in words of the array specified on the EVLABA=,addr parameter. If EVLABL=,n is omitted, the array is assumed to be 10 words.
$\text{`UBVLABA='}, \text{addr}$	Address of the array containing the user-specified beginning-of-volume labels that Q5REELSW is to write. The array must be on a word boundary. If UBVLABA=,addr is omitted, Q5REELSW does not write additional beginning-of-volume labels.
$\text{`UBVLABL='}, \text{n}$	Length in words of the array specified on the UBVLABA=,addr parameter. If UBVLABL=,n is omitted, the array is assumed to be 10 words.

Figure 9-38. Q5REELSW Call Format (Sheet 1 of 2)

Calling Parameters

'UEVLABA=',addr Address of the array containing the user-specified end-of-volume labels that Q5REELSW is to write. The array must be on a word boundary. If UEVLABA=,addr is omitted, Q5REELSW does not write additional end-of-volume labels.

'UEVLABL=',n Length in words of the array specified on the UEVLABA=,addr parameter. If UEVLABL=,n is omitted, the array is assumed to be 10 words.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'RBVLABL=',n Number of words written in the array specified on the BVLABA=,addr parameter (integer).

'REVLABL=',n Number of words written in the array specified on the EVLABA=,addr parameter (integer).

'STATUS=',stat Status code. Possible values: 0 through 199, 1475 through 1476, and 1627.

Figure 9-38. Q5REELSW Call Format (Sheet 2 of 2)

The caller must reissue any uncompleted tape functions interrupted when the end of volume was encountered.

Q5REELSW can also write additional tape volume labels and return all volume labels written.

A Q5REELSW call can specify arrays containing additional end-of-volume and beginning-of-volume labels to be written. Q5REELSW writes the end-of-volume labels after the EOVL label. It writes the beginning-of-volume labels after the BOVL label. The valid end-of-volume labels are EOVL through EOVL9 and UTL. The valid beginning-of-volume labels are UVL, HDR2 thru HDR9, and UHL. The label formats are shown in appendix F of this manual.

A Q5REELSW call can also specify arrays in which Q5REELSW returns the labels written (both system specified and user specified). If an array is specified, the call should also specify the RBVLABL or REVLABL parameter in which Q5REELSW returns the number of bytes written in the array.

Q5RETFIT - RETURN FIT

Call the Q5RETFIT routine (refer to figure 9-39) to return a FIT. The file associated with the FIT must be closed before its FIT is returned. Generate a new FIT for the file before SIL can again read or write the file.

The RETFIT parameter on the Q5CLOSE or Q5RETURN call can also return the FIT. SIL returns all FITs when the task completes.

No more than 110 FITs can be concurrently associated with a task.

<u>Call Format</u>	
CALL Q5RETFIT({ ^LFN= ,lfn ^FLUN= ,flun }, optional parameters)
<u>Calling Parameters</u>	
^LFN= ,lfn	File name. LFN=,lfn must be specified if FLUN=,flun is omitted.
^FLUN= ,flun	Logical number SIL assigned to the file. FLUN=,flun must be specified if LFN=,lfn is omitted.
<u>Return Parameters</u>	
^ERRLEN= ,len	Error message length in bytes (integer).
^ERRMSG= ,msg	Error message. The variable msg must be 80 bytes long.
^STATUS= ,stat	Status code. Possible values: 0 through 199, 253, 254, 303, 1424.

Figure 9-39. Q5RETFIT Call Format

Q5RETURN - RETURN FILE

Call the Q5RETURN routine (refer to figure 9-40) to return a file.

A file must be closed before it is returned.

A returned local file no longer exists; SIL releases its mass storage space. A returned permanent file is detached from the job.

Returning a file does not return the file's FIT unless the RETFIT parameter is specified.

Q5RETURN issues the DESTROY FILE system message.

Call Format

```
CALL Q5RETURN( { 'LFN=',lfn } , optional parameters )
               { 'FLUN=',flun }
```

Calling Parameters

'LFN=',lfn	File name. LFN=,lfn must be specified if FLUN=,flun is omitted.
'FLUN=',flun	Logical number SIL assigned to the file. FLUN=,flun must be specified if LFN=,lfn is omitted.
'RETFIT'	Indicates that SIL should discard the FIT for the file. If RETFIT is omitted, SIL does not discard the FIT.

Calling Parameter for Tape Files Only

'DRC'	Decrement resource count option. If DRC is specified, Q5RETURN decrements the tape drive reservation count specified on the RESOURCE statement for the job. If DRC is omitted, Q5RETURN does not decrement the reservation count.
'UL=',ul	Unload tape option. This option allows you to control the disposition of a tape after the file associated with it has been returned. Yes The tape will be unloaded No The tape will not be unloaded If the UL=ul option is omitted, the tape is unloaded in accordance with the IU option specified on the call to Q5RQUEST. If UL=,ul is specified, it overrides the IU option.

Figure 9-40. Q5RETURN Call Format (Sheet 1 of 2)

Return Parameters

'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	Error message. The variable msg must be 80 bytes.
'STATUS=',stat	Status code. Possible values: 0 through 199, 203, 204, 205, 254, 303, 1402, 1472, 1687, 1690.

Figure 9-40. Q5RETURN Call Format (Sheet 2 of 2)

Tape Files

Returning a tape file returns the file index entry for the file. If the specified file name is the name of a multifile set, Q5RETURN returns all files belonging to the set.

If the Q5RETURN call specifies the DRC parameter, Q5RETURN decrements the tape drive reservation count specified on the RESOURCE statement for the job.

If a Q5RETURN call specifies a tape parameter for a nontape file, Q5RETURN returns a warning error (status code 1472).

Q5REWIND - REWIND FILE

Call the Q5REWIND routine (refer to figure 9-41) to position a file at the beginning of its information.

The file must be open for read, write, append, or modify access.

If the file is an R, W, or L format file and the last operation SIL performed on the file was a write operation, Q5REWIND writes a file delimiter before rewinding the file.

Q5REWIND sets the record count in the FIT to zero.

If the file specified on a Q5REWIND call is a file connected to a terminal, control is returned to the caller with no action taken.

<u>Call Format</u>	
CALL Q5REWIND({ `LFN=` ,lfn `FLUN=` ,flun } , optional parameters)
<u>Calling Parameters</u>	
`LFN=` ,lfn	File name. LFN=,lfn must be specified if FLUN=,flun is omitted.
`FLUN=` ,flun	Logical number SIL assigned to the file. FLUN=,flun must be specified if LFN=,lfn is omitted.
<u>Return Parameters</u>	
`ERRLEN=` ,len	Error message length in bytes (integer).
`ERRMSG=` ,msg	Error message. The variable msg must be 80 bytes long.
`STATUS=` ,stat	Status code. Possible values: 0 through 199, 204, 206, 207, 253, 254, 303, 1405, 1416, 1580, 1596.

Figure 9-41. Q5REWIND Call Format

Tape Files

Q5REWIND positions a tape file at the beginning of its information. If the file is an ANSI labeled tape file, its beginning of information is immediately after its HDR1 label.

If the file is multivolume file and the current volume is not the first volume of the file, Q5REWIND prompts the operator to mount the first volume of the file and positions the volume at its beginning.

If the last I/O operation before the Q5REWIND call was a write operation, any data not yet written to the tape is written before the file is rewound. Q5REWIND writes the end-of-file indicator for the file before rewinding the file. If the file is an ANSI labeled tape, it writes the EOF1 label.

Q5ROUTE - ROUTE FILE

Call the Q5ROUTE routine (refer to figure 9-42) to specify a file disposition or to set certain file characteristics in the file index. Q5ROUTE can specify one of the following file dispositions:

- Discarding the file at task termination
- Storing the file as an unattached permanent file

Only the file owner or a privileged user can route a file. The file must be a local file or an attached permanent file. It becomes an unattached permanent file at its destination. Its FIT is not destroyed. Q5ROUTE may not be used to send a file to a remote system.

Read access permission for the file is required to route a file.

DEFER should always be specified when using Q5ROUTE to set file characteristics. If the DEFER parameter is specified on the Q5ROUTE call, SIL stores the file disposition in the file index entry but does not route the file. The file can be routed later with another Q5ROUTE call that specifies the file name but does not specify the DEFER parameter.

Q5ROUTE cannot route a file connected to a terminal. An attempt to do so returns a fatal error (status code 1463).

Q5ROUTE issues the FILE DISPOSITION system message.

Call Format

```
CALL Q5ROUTE( { ^LFN=',lfn  
               ^FLUN=',rflun } ,optional parameters)
```

Calling Parameters

^LFN=',lfn	File name. LFN=',lfn must be specified if FLUN=',rflun is omitted.
^FLUN=',rflun	Number SIL assigned to the file. FLUN=',rflun must be specified if LFN=',lfn is omitted.
^CM=',cm	Conversion mode. If CM=',cm is omitted, SIL uses the system default value.
^BI`	Binary
^DI`	Display code (64-character set)
^EC`	Extended display code (128-character set)

Figure 9-42. Q5ROUTE Call Format (Sheet 1 of 3)

Calling Parameters

`^DC=^,dc` Disposition code. If DC=,dc is omitted, Q5ROUTE uses the default value defined during system installation.

- `^IN^` Input to the input queue of the front-end system.
- `^LR^` Print on a 580-12 printer.
- `^LS^` Print on a 580-16 printer.
- `^LT^` Print on a 580-20 printer.
- `^PR^` Print on any available line printer.
- `^PU^` Punch file.
- `^P1^` Print on a 501 printer.
- `^P2^` Print on a 512 printer.
- `^SC^` Discard the file at the end of the task.

`^DEFER^` Indicates that file disposition is to be deferred. If DEFER is omitted, SIL performs the file disposition immediately.

`^DI=^,di` Eight ASCII characters (from the display code 64-character set) to be printed on the banner page at the front-end processor. If DI=,di is omitted, SIL uses the system default value.

`^EC=^,ec` Print or punch format. If EC=,ec is omitted, SIL uses the system default value.

- `^26^` 026 keypunch
- `^29^` 029 keypunch
- `^80^` 80-column binary

Files printed at the front-end processor use the following values:

- `^A4^` ASCII 48-character set
- `^A6^` ASCII 64-character set
- `^A9^` ASCII 95-character set
- `^B4^` BCD 48-character set
- `^B6^` BCD 64-character set

Figure 9-42. Q5ROUTE Call Format (Sheet 2 of 3)

Calling Parameters

`^IC=^,ic` File format. If `^IC=^,ic` is omitted, SIL uses the system default value.

`^AS^` 8-bit ASCII code; ANSI carriage control if print file

`^BI^` Binary

`^PA^` 8-bit ASCII code; ASCII carriage control if print file

`^OQNAME=^,oq` Five characters that identify the file in the output queue. The first character must be a letter. The system adds two unique job sequence characters as the sixth and seventh characters. The eighth character is a blank. If `OQNAME=,oq` is omitted, SIL uses the system default value.

`^ST=^,sid` Site identifier identifying the remote host associated with this file. If `ST=,sid` is omitted, SIL uses the system default value.

`^TID=^,tid` Terminal identifier. `tid` is a one- to seven-character user number of a logged-in remote user. For files destined for the central site (not a terminal), `tid` is zero. If `TID=,tid` is omitted, SIL uses the system default value.

Return Parameters

`^ERRLEN=^,len` Error message length in bytes (integer).

`^ERRMSG=^,msg` Error message. The variable `msg` must be 80 bytes long.

`^STATUS=^,stat` Status code. Possible values: 0 through 199, 204, 205, 303, 1425, 1463, 1690, 1700 through 1703, 1723.

Figure 9-42. Q5ROUTE Call Format (Sheet 3 of 3)

Q5RQUEST - REQUEST LOCAL FILE

Call the Q5RQUEST routine (refer to figure 9-43) to create a local file.

The Q5RQUEST call specifying a file must be the first reference to the file within a task. A FIT cannot exist for the file before the Q5RQUEST call.

A Q5RQUEST call sets the close file position in the FIT to no rewind. To change the value in the FIT, issue a Q5SETFIT call after the Q5RQUEST call.

Q5RQUEST cannot create a file having a security level higher than that of the task. An attempt to do so returns a message stating that the file is not found.

Q5RQUEST issues the CREATE system message.

Files Connected to Terminals

A Q5RQUEST call to request a file connected to an interactive terminal must specify TE as the device type (DT=,TE). Q5RQUEST cannot create a file connected to a terminal from within a batch job. An attempt to do so returns a fatal error (status code 1464).

If the Q5RQUEST call to request a file connected to a terminal specifies parameters valid only for mass storage or tape files, Q5RQUEST returns a warning error (status code 1454).

Tape Files

The Q5RQUEST call must specify a file name and the array containing the VSN list for the file. The VSN list contains the volume serial number of each tape volume assigned to the file. A file can comprise 1 through 255 volumes.

A Q5RQUEST call creates a system tapes table entry as well as a file index entry for the file.

The specified file can be a multifile set. If so, the file name is specified as the multifile set name on the Q5LABEL call for each file in the set. If the file is a multifile set, it must be an ANSI labeled tape file. The values specified on the Q5RQUEST call apply to each file in the multifile set.

If the tape volumes in the file have existing ANSI labels, the character conversion mode and density specified on the call must not conflict with the values used when the labels were written.

Call Format

CALL Q5RQUEST(ˆLFN=ˆ,lfn, optional parameters)

Calling Parameters

ˆLFN=ˆ,lfn File name. This parameter is required.

ˆACS=ˆ,acs Access permission set (any combination of the following letters without separators).

R	Read permission
W	Write permission
A	Append permission
M	Modify permission
X	Execute permission

If ACS=,acs is omitted, the access permission set in the file's FIT is used. If Q5RQUEST creates the FIT and ACS=,acs is omitted, ˆRWAMXˆ is assumed if the file is a mass storage file, ˆRˆ if the file is a tape file, and ˆRWˆ if the file is a file connected to a terminal.

ˆAU=ˆ,blocks Allocation unit. The integer number of 512-word blocks to be allocated when the file is extended. The value range of blocks is 1 to 65,535. If the file is created and blocks is not a multiple of the DAU for the device in which the first allocation occurs, blocks is rounded up to the next multiple of the DAU.

ˆDT=ˆ,dt Device type on which the file is to reside. DT=,dt is ignored if the file exists. Only DT=,MS is valid for a direct access file. If DT=,dt is omitted and no tape only parameters are specified, the file resides on mass storage. If tape only parameters are specified, the file resides on tape.

ˆMSˆ	Mass storage
ˆNTˆ	Magnetic tape
ˆTEˆ	Interactive terminal

ˆMNR=ˆ,mnr Minimum record length in bytes. For record formats other than F, SIL checks to see that the record is not shorter than this value. SIL does not use this parameter when writing F records. If MNR=,mnr is omitted, SIL assumes that the minimum record length is 1 byte.

ˆMXR=ˆ,mxr Maximum record length in bytes. For F-format records, mxr is the fixed record length. For other record formats, SIL checks to ensure that the record is not longer than this value. If MXR=,mxr is omitted, SIL assumes that the maximum record length is the default set by an installation parameter.

Figure 9-43. Q5RQUEST Call Format (Sheet 1 of 6)

Calling Parameters

`^SLEV=^,sl` Security level (1 through 8, but less than or equal to that of the calling task). If `SLEV=,sl` is omitted, SIL sets the file security level equal to that of the calling task.

Calling Parameters for Mass Storage Files Only

`^CT=^ ,ct` Communication type. If `^CT=^,ct` is omitted, the file is a non-RHF file.

`^RHF^` Remote Host Facility file

`^NRHF^` Non-Remote Host Facility file

`^RWF^` Remote Workstation Facility file

`^EXT=^,ext` File extendability indicator. If `EXT=,ext` is omitted, the file is extendable.

`^Y^` The file is extendable

`^N^` The file is not extendable

`^FC=^,fc` File category. If `FC=,fc` is omitted, the file is a user file.

`^B^` Batch input file

`^U^` User file

`^LEN=^,fl` File length in 512-word blocks. If `LEN=,fl` is omitted, the file is eight 512-word blocks.

`^NOSEG^` Indicates that file must be contiguous (not written in segments). If `NOSEG` is omitted, SIL can segment the file.

`^PN=^,pn` Six-character identifier of a disk pack in the device set on which SIL creates the file. If `PN=,pn` is omitted, the system assigns mass storage space for the file.

`^SFO=^,fo` File organization. If `SFO=,fo` is omitted, SIL assumes the installation-defined default organization (released value, sequential access).

`^D^` Direct access

`^S^` Sequential access

`^TYPE=^,typ` File type. If `TYPE=,typ` is omitted, SIL assumes that the file is a physical data file.

`^PD^` Physical data file

`^VC^` Virtual code (controllee) file

Figure 9-43. Q5RQUEST Call Format (Sheet 2 of 6)

Calling Parameters for Mass Storage and Tape Files Only

`^BT=^,bt` Blocking type. If `BT=,bt` is omitted, the file has character count blocking.

`^C^` Character count blocking

`^I^` Internal blocking

`^K^` Record count blocking

`^PC=^,pc` Padding character for F records. If `PC=,pc` is omitted, SIL pads with the installation-defined character (released value, blank).

`^RMK=^,rmk` Record delimiting character for R-type records. If `RMK=,rmk` is omitted, SIL uses the installation-specified character [usually ASCII US (#1F)].

`^RT=^,rt` Record format. If `SFO=,D` is specified, the only valid record format is F. If `RT=,rt` is omitted, SIL assumes the installation default format (released value, R) for sequential access files and F format for direct access files.

`^B^` System block (for tape files only)

`^F^` ANSI fixed length

`^L^` CYBER Record Manager control word (for tape files only)

`^R^` Record mark delimited

`^U^` Undefined

`^W^` Control word delimited

Calling Parameters for Tape Files Only

`^CCS=^,cm` Character conversion mode. If the tape is labeled, the conversion mode must match the conversion mode used when the labels were written. Conversion is performed only if the data conversion option (CONV) is specified on the tape request for the file.

`^AS^` ASCII character set

`^EB^` EBCDIC character set

If `CM=cm` is omitted and the tape is labeled, the default mode is the conversion mode used when the labels were written. If `CM=cm` is omitted and the tape is unlabeled, the installation-defined default mode (released value, ASCII) is used.

Figure 9-43. Q5RQUEST Call Format (Sheet 3 of 6)

Calling Parameters for Tape Files Only

`^CONV^` Data conversion option. If CONV is specified, tape data is read and written as character codes, using the character set specified by the CM parameter. If CONV is omitted, no conversion is performed and the data is read and written as binary data.

`^DEN=^,den` Recording density.

`^PE^` 1600 cpi

`^GE^` 6250 cpi

If DEN=,den is omitted, the installation-defined default density (released value, 6250 cpi) is used.

`^HEC^` Hardware error correction option for GCR tapes (6250 cpi). If HEC is specified and the site has enabled the option, the system allows the writing of single-track errors that can be corrected as the tape is read (on-the-fly correction). If HEC is omitted, the system performs standard error recovery for single-track errors.

`^IU^` Inhibit unload option indicating whether the system unloads a tape volume when its file is returned. If IU is specified, the system does not unload the tape volume. If IU is omitted, the system unloads the tape volume.

`^LABA=^,adr` Address of a 160-byte array containing the new VOL1 and HDR labels for the tape volume. The array must begin on a word boundary. To write the labels, the call must specify LPROC=,W. If LABA=,adr is omitted, no labels are written.

`^LABEL=^,lsl` Indicates the labels read or written.

`^AN^` ANSI standard labels

`^NS^` Nonstandard labels

`^U^` Unlabeled file

If LABEL=,lsl is omitted, the file is assumed to be an ANSI standard labeled file.

`^LPROC=^,lp` Label processing option.

`^R^` Read existing labels (verify existing HDR1 label).

`^W^` Write new labels.

If LPROC=,lp is omitted, label processing depends on the value of the ACS parameter. For ACS=,R or ACS=,RW, LPROC=,R is assumed. For ACS=,W, LPROC=,W is assumed.

Figure 9-43. Q5RQUEST Call Format (Sheet 4 of 6)

Calling Parameters for Tape Files Only

<code>^MPRU=</code> ,mpru	MPRU size in bytes; used only if the file uses the V tape format. If MPRU=,mpru is omitted, the MPRU size for V format is 32768 bytes.
<code>^NEOI=</code> ,neoi	No EOI detection option indicating whether an EOI status is returned if two consecutive tape marks are encountered. This option is valid only for unlabeled, non-ANSI variable format tapes (<code>^TF=</code> , <code>^V</code> , <code>^LABEL=</code> , <code>^U</code> on the REQUEST line). If NEOI is specified, the system does not return EOI status. If NEOI is not specified, the system returns an EOI status when two consecutive tape marks are encountered.
<code>^NS=</code> ,ns	Noise size in bytes; this is used only while reading tapes generated on a non-CYBER 200 system. Default is zero. $0 \leq ns \leq 31$.
<code>^OMSGA=</code> ,adr	Address of array containing a message to be displayed on the operator's 0 display. If OMSGA=,adr is omitted, no message is displayed. If OMSGA= is specified, OMSGL= must also be specified. The message appears in the 0 display after the system mount message.
<code>^OMSGL=</code> ,n	Number of bytes in the operator message. If OMSGL=,n is omitted, no message is displayed. If OMSGL=,n is specified, OMSGA=,adr must also be specified. The value n must be supplied and be between 1 and 64 inclusive. No default for n is assumed.
<code>^OVA=</code> ,va	Volume accessibility character in the existing VOL1 label. This parameter is required only if the character is nonblank.
<code>^RA=</code> ,ra	Ring access option. This parameter cannot be specified if ACS= is specified. When <code>^RA=</code> is specified, if the tape is already mounted with the ring in, Read/Write access is assigned; otherwise, Read only access is assigned.
<code>^RPB=</code> ,rpb	Records per block; used only for the K blocking type. If RPB=,rpb is omitted, one record per block is assumed.
<code>^RU</code>	Read unconditional option indicating whether the system allows reads past the end-of-tape (EOT) reflective marker on the tape volume. If RU is specified, the system allows reads past the EOT. If RU is omitted, the installation-defined default option is used (released value, no reads past EOT).
<code>^TF=</code> ,fmt	Tape data format. <code>^I</code> NOS internal format <code>^SI</code> SCOPE internal format <code>^LB</code> Large block format <code>^V</code> Variable block format <code>^NV</code> Non-ANSI, variable If TF=,fmt is omitted, the installation-defined default format (released value, LB) is used.

Figure 9-43. Q5RQUEST Call Format (Sheet 5 of 6)

Calling Parameters for Tape Files Only

'VSN=',adr Address of the array containing the VSN list. The VSN list is the list of tape volumes associated with the file name. This parameter is required for a tape file.

 Q5RQUEST returns a volume serial number in each word of the array. The volume serial number is returned as six ASCII characters in the rightmost 6 bytes of the word; the leftmost 2 bytes are blank-filled.

'VSNL=',n Number of words in the VSN list array. If VSNL=,n is omitted, 1 is assumed.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'RFLUN=',rflun File logical unit number SIL assigned to the FIT (integer).

'STATUS=',stat Status code. Possible values: 0 through 199, 203, 204, 205, 253, 1453, 1464, 1469, 1472, 1490 through 1493, 1495, 1497, 1505, 1515, 1524, 1637, 1685, 1688, 1710, 1711, 1716, 1720, 1722, 1725, 1762, 1765 through 1772.

Return Parameters for Mass Storage Files Only

'CONT=',con Initial contiguity of the mass storage file (ASCII, left-justified, blank filled).

 Y The file space is contiguous.

 N The file space is not contiguous.

'RPN=',pn Six-character identifier of the disk pack on which the file resides.

Figure 9-43. Q5RQUEST Call Format (Sheet 6 of 6)

Q5SETFIT - SET FIT FIELD VALUES

Call the Q5SETFIT routine (refer to figure 9-44) to change the values of specified FIT fields. The FIT must already exist. The values in the fields not specified are not changed.

<u>Call Format</u>	
CALL Q5SETFIT({ ^LFN=,lfn ^FLUN=,rflun } ,optional parameters)
Calling Parameters	
^LFN=,lfn	File name in the FIT. LFN=,lfn must be specified if FLUN=,rflun is omitted.
^FLUN=,rflun	Number SIL assigned to the FIT. FLUN=,rflun must be specified if LFN=,lfn is omitted.
^ACS=,acs	Access permission set (any combination of the following letters without separators).
	R Read permission
	W Write permission
	A Append permission
	M Modify permission
	X Execute permission
	If ACS=,acs is omitted, the access permission set in the FIT is not changed.
^WSA=,wsa	Working storage area used by get and put calls. If WSA=,wsa is omitted, the working storage area is not changed.
^WSL=,wsl	Length (in bytes) of the working storage area. If WSL=,wsl is omitted, the working storage area length is not changed.
<u>Calling Parameters for Mass Storage Files Only</u>	
^BN=,bn	Number of the next available block for reading or writing. If BN=,bn is omitted, the block number is not changed.
^SFO=,fo	File organization. If SFO=,fo is omitted, the file organization is not changed.
	^D Direct access organization
	^S Sequential organization

Figure 9-44. Q5SETFIT Call Format (Sheet 1 of 4)

`^TRY=^,try` Error retry option indicating whether, if a data error is detected, the system performs its standard error recovery procedures.

`^NR^` Error recovery not performed.

`^SR^` Error recovery performed.

If `TRY=,try` is omitted, error recovery is performed.

Calling Parameters for Tape Files Only

`^VSN=^,adr` Address of the array containing the VSN list. The VSN list is the list of tape volumes associated with the file name. This parameter is required for a tape file.

Q5RQUEST returns a volume serial number in each word of the array. The volume serial number is returned as six ASCII characters in the rightmost 6 bytes of the word; the leftmost 2 bytes are blank-filled.

`^VSNL=^,n` Number of words in the VSN list array. If `VSNL=,n` is omitted, 1 is assumed.

Return Parameters

`^ERRLEN=^,len` Error message length in bytes (integer).

`^ERRMSG=^,msg` Error message. The variable `msg` must be 80 bytes long.

`^RFLUN=^,rflun` File logical unit number SIL assigned to the FIT (integer).

`^STATUS=^,stat` Status code. Possible values: 0 through 199, 203, 204, 205, 253, 1453, 1464, 1469, 1472, 1490 through 1493, 1495, 1497, 1505, 1515, 1524, 1637, 1685, 1688, 1710, 1711, 1716, 1720, 1722, 1725, 1762, 1765 through 1772.

Return Parameters for Mass Storage Files Only

`^CONT=^,con` Initial contiguity of the mass storage file (ASCII, left-justified, blank filled).

Y The file space is contiguous.

N The file space is not contiguous.

`^RPN=^,pn` Six-character identifier of the disk pack on which the file resides.

Return Parameters for Tape Files Only

`^RACS=^,racs` Access permission set assigned to the tape file by the system (ASCII, left justified, blank filled).

R Read permission.

W Write permission.

R/W Read and Write permission.

Figure 9-43. Q5RQUEST Call Format (Sheet 6 of 6)

Calling Parameters for Mass Storage and Tape Files Only

`^OFP=^,ofp` File positioning when the file is opened. If `OFP=,ofp` is omitted, the open position is not changed.

- `^R^` Rewind the file.
- `^P^` Position the file after the last block read or written (tape files only).
- `^N^` Do not rewind the file.

`^PC=^,pc` Padding character for F format records. If `PC=,pc` is omitted, the padding character is not changed.

`^RMK=^,rmk` Record delimiting character for R format records. If `RMK=,rmk` is omitted, the record delimiter is not changed.

`^RT=^,rt` Record type. If `RT=,rt` is omitted, the record type is not changed.

- `^B^` System block
- `^F^` ANSI fixed length
- `^L^` CYBER record manager control word
- `^R^` Record mark delimited
- `^U^` Undefined
- `^W^` Control word

`^SRF=^,srf` Indicates that SIL must complete an I/O request before returning control to the caller. If `SRF=,srf` is omitted, the I/O overlap option is not changed.

- `^Y^` Suppress overlapped I/O.
- `^N^` Allow overlapped I/O.

Calling Parameters for Tape Files Only

`^ADO=^,ado` Assembly/disassembly option used for CYBER 170/CYBER 200 binary tape interchange (ASCII, left justified, blank filled). If selected, each 60 bits of tape data read is stored in a CYBER 200 64-bit word with the upper 4 bits as zero.

- `^BI^` Binary; no assembly/disassembly performed.
- `^BW^` 60 to 64; assembly/disassembly performed.

Figure 9-44. Q5SETFIT Call Format (Sheet 3 of 4)

Calling Parameters for Tape Files Only

NOTE

The assembly/disassembly option (ADO=,BW) is invalid if the data conversion option is selected (CONVERT). Assembly/ disassembly is valid only for binary data; it is not valid for character coded data.

^CM=^,cm Character conversion mode.

 ^AS^ ASCII character set

 ^EB^ EBCDIC character set

Return Parameters

^ERRLEN=^,len Error message length in bytes (integer).

^ERRMSG=^,msg Error message. The variable msg is 80 bytes long.

^RFLUN=^,rflun Number SIL assigned to the FIT.

^STATUS=^,stat Status code. Possible values: 0 through 199, 253, 254, 262, 303, 1454, 1650.

Figure 9-44. Q5SETFIT Call Format (Sheet 4 of 4)

Q5SKIP - SKIP PARTITION

Call the Q5SKIP routine (refer to figure 9-45) to reposition a file.

To reposition a file, read access permission to the file is required.

The file cannot be repositioned forward if the last I/O operation was a write operation.

For a direct access file, Q5SKIP can only skip records. It increments or decrements the record count in the FIT by the specified count.

For a U format file, Q5SKIP can only skip blocks.

If the file is empty, an attempt to reposition the file returns an error (warning code 1414 for a backward skip, fatal code 1434 for a forward skip).

If the file specified on a Q5SKIP call is a file connected to a terminal, control is returned to the caller with no action taken.

Call Format

```
CALL Q5SKIP( { ^LFN=,lfn  
              ^FLUN=,flun } ,optional parameters)
```

Calling Parameters

^LFN=,lfn	File name. LFN=,lfn must be specified if FLUN=,flun is omitted.
^FLUN=,flun	Logical number SIL assigned to the file. FLUN=,flun must be specified if LFN=,lfn is omitted.
^COUNT=,cnt	Number of partitions to skip. If the number is a negative value, SIL backspaces the file the requested number of partitions. If COUNT=,cnt is omitted, SIL skips forward one partition.
^PART=,part	Partition type to be skipped. This parameter is ignored for direct access files. If PART=,part is omitted, Q5SKIP skips records.
^R	Record
^G	Group
^B	Block (PRU for tape files)
^L	Logical record unit (LRU) (tape files only)
^F	File

Figure 9-45. Q5SKIP Call Format (Sheet 1 of 2)

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).
'ERRMSG=',msg Error message. The variable msg is 80 bytes long.
'STATUS=',stat Status code. Possible values: 0 through 199, 204, 206, 207, 253, 254, 303, 1405, 1412 through 1416, 1422, 1434, 1438, 1439, 1441, 1444, 1472, 1476, 1489, 1498, 1580, 1596, 1726.

Return Parameters for Tape Files Only

'RCOUNT=',n Number of partitions skipped (integer). Q5SKIP returns the count after its repositioning is complete. If RCOUNT=,n is omitted, Q5SKIP does not return the number of partitions skipped.
'RSN=',rsn Request serial number that uniquely identifies the request (integer). If RSN=,rsn is specified, control returns to you before the tape file repositioning is complete. To determine whether the repositioning has completed, specify the RSN on a Q5CHECK call.

If RSN=,rsn is omitted, Q5SKIP completes its repositioning of the file before returning control to the you.

Figure 9-45. Q5SKIP Call Format (Sheet 2 of 2)

Repositioning a Sequential Access File

For a sequential access file, Q5SKIP changes the current file position the specified number of partitions forward or backward. The specified partitions can be records, groups, files, or blocks.

At completion of a Q5SKIP call, the file is positioned immediately after the specified partition delimiter except when skipping backward by files. A request to skip backward one partition positions the file immediately before the file delimiter.

If a backward skip is requested on an R, L, or W format file and the last I/O operation SII performed on the file was a write operation, Q5SKIP writes the file delimiter used by the record format before repositioning the file.

Call Q5GETFIT to determine the current file position.

Example of Sequential Access File Positioning

For example, assume that an R format file named FILE1 has the following structure:

```
      record      record      file
data...delimiter...data...delimiter...delimiter
```

Assume that a task opens the file so that its current file position is at the beginning of information (BOI) as indicated by the following caret character:

```
      record      record      file
^data...delimiter...data...delimiter...delimiter
```

Assume that the task executes the following call to skip forward two records.

```
Q5SKIP('LFN=', 'FILE1', 'COUNT=', 2)
```

Q5SKIP positions the file after the second record delimiter but before the file delimiter. [The task could append data at this position (at EOF).]

```
      record      record      file
data...delimiter...data...delimiter^delimiter
```

The task then executes the following call to skip backward one record.

```
Q5SKIP('LFN=', 'FILE1', 'COUNT=', -1)
```

Q5SKIP positions the file after the first record delimiter (at BOR).

```
      record      record      file
data...delimiter^data...delimiter...delimiter
```

Next, the task executes the following call to skip to the end of the file.

```
Q5SKIP('LFN=', 'FILE1', 'PART=', 'F')
```

Q5SKIP positions the file after the file delimiter (at EOI).

```
      record      record      file
data...delimiter...data...delimiter...delimiter^
```

To position the file before the file delimiter so that it can append data, the task could next execute either of the following calls:

```
Q5SKIP('LFN=', 'FILE1', 'COUNT=', -1)
Q5SKIP('LFN=', 'FILE1', 'PART=', 'F', 'COUNT=', -1)
```

In either case, Q5SKIP skips backward past the file delimiter (at EOF).

```
      record      record      file
data...delimiter...data...delimiter delimiter^
```

This chapter contains information for a system programmer who is interested in developing an application or utility.

CONVENTIONS

A controllee execute line is entered for processing by VSOS either as a batch processor control statement or as an interactive terminal type-in. An execute line can occur as one or more physical records representing card images or terminal lines. From the point of view of the common execute line supporting routines, an exact correspondence exists between batch commands and terminal commands, including continuation of the command text to more than one card or terminal line. (From the point of view of the user, however, this correspondence does not exist.)

Standard processing is done on five types of linguistic expressions called tokens. The tokens are:

- Alphanumeric identifiers.
- Decimal numeric constants.
- Hexadecimal numeric constants prefixed by the character #.
- Character or string constants delimited by the character ".
- The special characters, which are / # " & .) , = and blank.

The # character is referred to in text as a hash mark. The & character is an ampersand.

Execute line options are defined by means of positional or keyword-identified values. Standard diagnostics are issued if abnormal syntax or conditions are encountered.

A set of four system library routines are to be used to guarantee adherence to the conventions previously stated. The routines are:

<u>Routine</u>	<u>Description</u>
Q7ENVIRN	Determines the program environment.
Q7KEYWRD	Processes the text of an execute line.
Q7MODE	Determines if the task's controller is a terminal.
Q7PROMPT	Provides interaction with the controller; collects parts from several input records, and builds the complete character string for processing by Q7KEYWRD.

When a controllee execute line requires more than one terminal line, an ampersand must be used to designate continuation to the subsequent line. Card image continuation is performed automatically during batch processing if a terminator character has not been encountered. The ampersand signals a logical end of record and can be followed by comments. The text of the execute line consists of two or more tokens: the first is alphanumeric and identifies the task name, while the last is a special character called a terminator. The terminator characters are a period and right parenthesis. An implicit terminator occurs at the end of a terminal line that does not contain an ampersand. Comments can be placed immediately following a terminator character or an ampersand. The following execute lines are equivalent:

SAMPLE,A. optional comments

SAMPLE&,A. optional comments

SAMPLE A

A parameter list can follow the task name but must precede the terminator character. Order-dependent parameters must be in the order specified; key-dependent parameters can appear in any order. Parameter formats depend on the control statement specified, but they always follow the same general guidelines.

Consecutive parameter list items are separated by level-1 separator characters comma and blank. In addition, the left parenthesis acts as a level-1 separator between the task name and the parameter list. A parameter list item can be defined by a list of user numbers or file names. These values are also separated by the level-1 parameter separators. A file name can be followed by attributes of disposition code or length, with attributes separated from each other by the level-2 separator character slash.

Blank is a special character and only performs a separator function when not used with other separators or terminators. Any level of separator can be preceded or followed by blanks, which serve only to highlight the separator; in a similar fashion, the terminator characters can be preceded by highlighting blanks.

System utilities or tasks provide default settings for all on/off options. In addition, the input, output, and binary file options have the default names INPUT, OUTPUT, and BINARY. Where tasks create files for the user, the task can determine the necessary file size or the user is allowed to submit an estimate of an adequate size. Tasks that create files also determine the disposition of the file upon task completion. The user has the opportunity to specify file disposition.

The task name is constructed of one to eight letters and digits. Except where reference is made to a drop file, the first character must be a letter. The task name is bound on the left by the start of the command and on the right by a level-1 separator or a terminator.

Order-dependent parameters are strings of nonseparator, nonterminator characters. Their interpretation is strictly a function of the particular product. An order-dependent parameter list is ended by a terminator or by the occurrence of a key-dependent parameter.

The following are examples of execute lines using order-dependent parameter lists:

```
COPY(FILEA,FILEB)
```

```
PURGE,FILE1,FILE2,FILE3.
```

A key-dependent parameter has the general structure shown in figure 10-1. The following is an example of an execute line using a key-dependent parameter list:

```
FTN(I=COMPILE,L=OUTPUT,B=BINARY/PU/#240)
```

key=defs	
key	A string of letters and numbers, 1 to 255 characters, delimited to the left by a level-1 separator and on the right by an = character, a separator, or a terminator.
defs	Strings of nonseparator, nonterminator characters whose interpretation is strictly a function of the particular product and the key identifier.

Figure 10-1. Key-Dependent Parameter Format

Examples of the use of both parameter forms are:

```
WXYZ(FILE1,FILE2,OU=MAPFILE)
```

To ensure that ambiguities do not arise, the programmer calling the keyword word processors must not allow the following:

- A parameter resembling a file name to follow a file name list unless that parameter has a key.
- A parameter resembling a user number to follow a user number list unless that parameter has a key.
- A parameter resembling a text string to follow a text string unless that parameter has a key.

Parameter values can be strings of letters and digits, decimal digit strings, hexadecimal digit strings, and character strings delimited by quotation marks. In some cases, the alphanumeric string can occur as two decimal digits followed by one to six letters and numbers. This exception is provided to accommodate drop file names. Decimal digit strings are normally interpreted as decimal constants; a hexadecimal constant is normally preceded by a hash mark. Values that must be virtual bit addresses are always hexadecimal values even if the hash mark is not present. In some cases, such as the GROS option of the loader, a hash mark is required to distinguish the address from identifier data in the same list. Some examples are:

```
WXYZ(EN="!FILE!",OU=MAP/#10)
```

```
WXYZ(FILE,OU=MAP/16,LI=SYSLIB,MYLIB)
```

```
PURGE,12DROP,JUNK.
```

```
COPY,42DROP,SAVEDROP.
```

Lists of values are as order-flexible as the values permit; the user is normally given maximum flexibility consistent with the task requirements. The following equivalent parameter strings illustrate this flexibility:

```
B=FILE/10/PR
```

```
B=FILE/PR/10
```

All key-dependent parameters have on and off settings where appropriate, and can be turned on and off. Turning on keys can be accomplished by means of a key=1 parameter, or by use of the key name only; these keys can also be turned off by means of a key=0 parameter. File identification keys should be turned off with key=0. Where the option is normally off, a parameter of the form key=filename turns the option on for a specific file, while use of the key name only turns the option on for a default file.

The following execute lines are equivalent, and illustrate the on/off ability:

```
IMPL,X.
```

```
IMPL,I,X=1.
```

```
IMPL,X,I=INPUT,B=BINARY/#40.
```

SUPPORTING ROUTINES

Common execute line standards are supported by four subroutines from the system library. The subroutines, which are callable from FORTRAN, META, and IMPL, are:

<u>Subroutine</u>	<u>Description</u>
Q7ENVIRN	Determines the program environment of the calling task. The task may be in one of three environments: batch, interactive, or no level-1 controller.
Q7MODE	Determines whether the parent controller of the calling task is a terminal or another task. A batch job falls into the latter category.
Q7PROMPT	Inputs parameters to be passed to Q7KEYWRD for syntax checking. It prompts terminal users for input if no parameters are specified in the execute line. It also strips the trailing period or matching outside parenthesis characters from the parameter text before calling Q7KEYWRD.
Q7KEYWRD	Examines a character string, checks its syntax, and converts data to internal format. In the case of a detected error, it prints error messages and requests; in interactive mode, it permits error correction by accepting reinput of an execute line parameter. Also, in interactive mode, Q7KEYWRD can be set to request and input each parametric keyword through the use of an appropriate prompting message.

Assembly language routines call any of these subroutines by using the FORTRAN or IMPL type of calling sequence, while FORTRAN and IMPL programs access the routines using CALL statements.

If the main program is coded in FORTRAN, the original execute line is processed by FORTRAN initialization for run-time file substitution. In interactive mode, the program may subsequently call Q7PROMPT or Q7KEYWRD for other lines.

Q7ENVIRN

The function of this subroutine is to determine a task's program environment and to return the information in a full word whose variable name is supplied as the only parameter to the Q7ENVIRN routine. A full word is defined as a 64-bit word that is aligned on a word boundary. The call statement format of Q7ENVIRN is shown in figure 10-2.

Q7ENVIRN (environ)	
environ	A full-word variable in which one of the following values is returned:
0	The task is executing from within a batch job.
1	The task is executing from within an interactive session.
2	The task does not have a level-1 controller (for example, QTF, PTFS, or QTFS).

Figure 10-2. Q7ENVIRN Call Statement Format

Q7MODE

The function of this subroutine is to determine if a task's controller is a terminal and to return the information in a full word whose variable name is supplied as the only parameter to the Q7MODE routine. A full word is defined as a 64-bit word that is aligned on a word boundary. The call statement format of Q7MODE is shown in figure 10-3.

```
CALL Q7MODE (mode)
```

mode A full-word variable in which one of the following values is returned:

0 Controller is not a terminal.

1 Controller is a terminal.

Figure 10-3. Q7MODE Call Statement Format

Q7PROMPT

This routine serves as an interface between a calling routine and the Q7KEYWRD subroutine. It inputs user parameters into an input buffer, then passes the text to Q7KEYWRD for syntax checking. If no text is specified on the execute line, Q7PROMPT can prompt the interactive user for parameters, using the message PLEASE SPECIFY PARAMETERS or a message provided by the calling routine; otherwise, if no text is specified on the execute line, it can proceed with a call to Q7KEYWRD, optionally setting bit 60 in the options parameter, which causes Q7KEYWRD to prompt for individual keywords.

An input buffer can either be supplied by the calling routine or allocated by Q7PROMPT. Delineator characters, such as matching outside parentheses or a trailing period, are deleted from the input text prior to the call to Q7KEYWRD.

Input text can be continued on succeeding lines in interactive mode, provided that an ampersand (continuation character) is appended to each line.

The call statement format of Q7PROMPT is shown in figure 10-4. The opt, r, rbuf, rlen, and t_i parameters are not used by Q7PROMPT, but are passed to Q7KEYWRD for use in syntax checking. If a text string is not specified in the controllee execute line and the p parameter is not negative, Q7PROMPT prompts for parameters and saves them in a buffer with the name specified as the buf parameter.

```
CALL Q7PROMPT (txt,p,opt,r,buf,blen,rbuf,rlen,t1,...,tn)
```

txt	Text string to be passed to Q7KEYWRD. The string must contain any desired carriage control characters.
p	Indicates whether prompting is desired: <ul style="list-style-type: none">>0 Number of character bytes in txt. Use txt to prompt for parameters.0 Use the text string PLEASE SPECIFY PARAMETERS to prompt for parameters.-1 Do not prompt for parameters. The value of the variable blen is 0.-2 Do not prompt for parameters. Options bit 60 should be 0.-3 Do not prompt for parameters. Wait for message.
buf	Name of buffer file into which the parameters are to be read. If the blen field is 0, the buf field is the name given to a buffer provided by Q7PROMPT.
blen	Name of a full-word variable whose nonzero value indicates the number of character bytes in buf. If the value of blen is 0, no buffer is provided by the caller; in this case, Q7PROMPT allocates a 4096-character buffer named buf. A count of the number of characters actually read is returned by the system into the blen field.

The opt, r, rbuf, rlen, and t_i fields are described under the Q7KEYWRD call statement.

Figure 10-4. Q7PROMPT Call Statement Format

Q7KEYWRD

The keyword subroutine scans a line of text, checks syntax, and converts data to internal formats. It prints error messages and inputs replacement expressions as required. Q7KEYWRD processes text containing both positional and keyword type parameters. The calling routine provides Q7KEYWRD with syntax tables that completely describe the general format of the input parameters. Q7KEYWRD uses the tables to interpret the specific parameters in the execute line test. These input parameters, called keyword expressions, are written as follows:

key₁ key₂ key₃ . . .

Each key_i is separated from other keyword expressions by one or more blanks or by commas, and has one of the following formats:

lhs = rhs

lhs

rhs

The syntax tables for each key_i keyword relate the valid left-hand sides (lhs) of the expression to valid right-hand sides (rhs). This includes specifying whether the degenerate cases, lhs and rhs, are to be treated as having no left-hand side or no right-hand side. Each key_i, then, can be any one of the following possibilities:

lhs(1) = rhs(1,1)
.
.
.
lhs(1) = rhs(1,n1)
lhs(2) = rhs(2,1)
.
.
.
lhs(m) = rhs(m,nm)

m is the number of possible left-hand sides for the expression, left-hand side k having nk possible right-hand sides.

The syntax tables also specify positional relationships among the keyword expressions. A given expression, key_i , can be flagged as positional, meaning that it must appear after expressions $key_1, \dots, key_{(i-1)}$ but before the expressions $key_{(i+1)} \dots$. If an expression (key_i) is not flagged as positional, it can appear in any order with preceding or succeeding nonpositional expressions; so, if $key_i, key_{(i+1)},$ and $key_{(i+2)}$ are nonpositional, any of the following are valid:

```
key(i), key(i+1), key(i+2)
key(i), key(i+2), key(i+1)
key(i+1), key(i), key(i+2)
key(i+1), key(i+2), key(i)
key(i+2), key(i), key(i+1)
key(i+2), key(i+1), key(i)
```

The syntax tables also indicate which key_i parameters are required in the execute line text. If a parameter flagged as required is not encountered in its required location, an error message is issued.

Left-hand sides for an expression include:

- None (the degenerate case, lhs).
- A literal character string, 1 to 255 characters long.

Right-hand sides for an expression include:

- None (the degenerate case, rhs).
- A literal character string, 1 to 255 characters long.
- An arbitrary character string, 1 to 255 characters long.
- Any remaining unscanned text, up to 255 characters maximum.
- A number in the range 0 to $2^{47}-1$ (table setting indicates whether the number can be decimal, hexadecimal with a leading # character, or an address in hexadecimal with no leading # sign required; table settings can also indicate the range of the number if the default range is not sufficiently restrictive).
- A user number.
- A file name (table settings indicate whether a drop file name can be specified and whether the length, print, and punch attributes can be specified).

Lists of numbers, user numbers, and file names separated by slashes, blanks, or commas can be allowed as right-hand sides. A field in the syntax tables indicates that lists are to be allowed and specifies the maximum number of elements permissible.

The entry point Q7KEYWRD is used for both FORTRAN and IMPL calling sequences. The call statement format of Q7KEYWRD is shown in figure 10-5. The lhs table pointers are illustrated in figure 10-6. Each entry in an lhs table points to an rhs table (also full-word-aligned) that describes valid right-hand sides for the given left-hand side and specifies the format in which information is returned in the return buffer to the calling routine.

```
CALL Q7KEYWRD(opt,r,buf,blen,rbuf,rln,t1,...,tn)
```

opt Name of a full-word variable, the rightmost 5 bits of whose value indicate the following options:

<u>Bit</u>	<u>Description</u>
59	If 0, send error message to the terminal. If set to 1, return error message to the caller.
60	If 0, scan the input for keyword expressions. If set to 1, prompt for each keyword listed in the tables t ₁ ,...,t _n .
61	If 0, or if user enters "cancel" in response to interactive prompt, abort on syntax error. If set to 1, return to caller on either condition.
62	If 0, prompt for replacement on syntax error. If set to 1, do not prompt for replacement.
63	If 0, send error messages to program controller for output. If set to 1, do not output error messages.

r Name of full-word variable to contain return codes. Return codes are:

0	Text scanned successfully.
1	Internal error; or parameters processed did not match any left-hand side or right-hand side tables; or user entered "cancel" in response to interactive prompt.
2	Return buffer too small.
3	Incorrect number of parameters in Q7 PROMPT/Q7 KEYWORD call line.
4	Invalid type field in lhs table entry.
5	Invalid type field in rhs table entry.
6	Invalid flags field in rhs table entry.
7	Words field for return buffer entry exceeds 255.
8	Options field bit 60 is 1, and prompt message length or address in lhs table header is 0.

Code 1 is returned only if bit 61 of opt field is 1.

Figure 10-5. Q7KEYWRD Call Statement Format (Sheet 1 of 2)

buf	Virtual bit address of string to be scanned for keyword expressions. This field is not used if prompting is requested (options bit 60 is 1).
blen	Name of full-word variable whose value specifies the number of characters in the string indicated by buf. This field is not used if options bit 60 is 1.
rbuf	Virtual bit address of the full-word-aligned buffer (the return buffer) in which reformatted keyword information is to be returned.
rlen	Name of full-word variable whose value specifies the number of characters in the return buffer.
t _i	Virtual bit address of full-word-aligned lhs table (figure 10-6) that describes acceptable syntax constructs and specifies formats for the returned information. The number of addresses varies with the syntax of the line being scanned.

Figure 10-5. Q7KEYWRD Call Statement Format (Sheet 2 of 2)

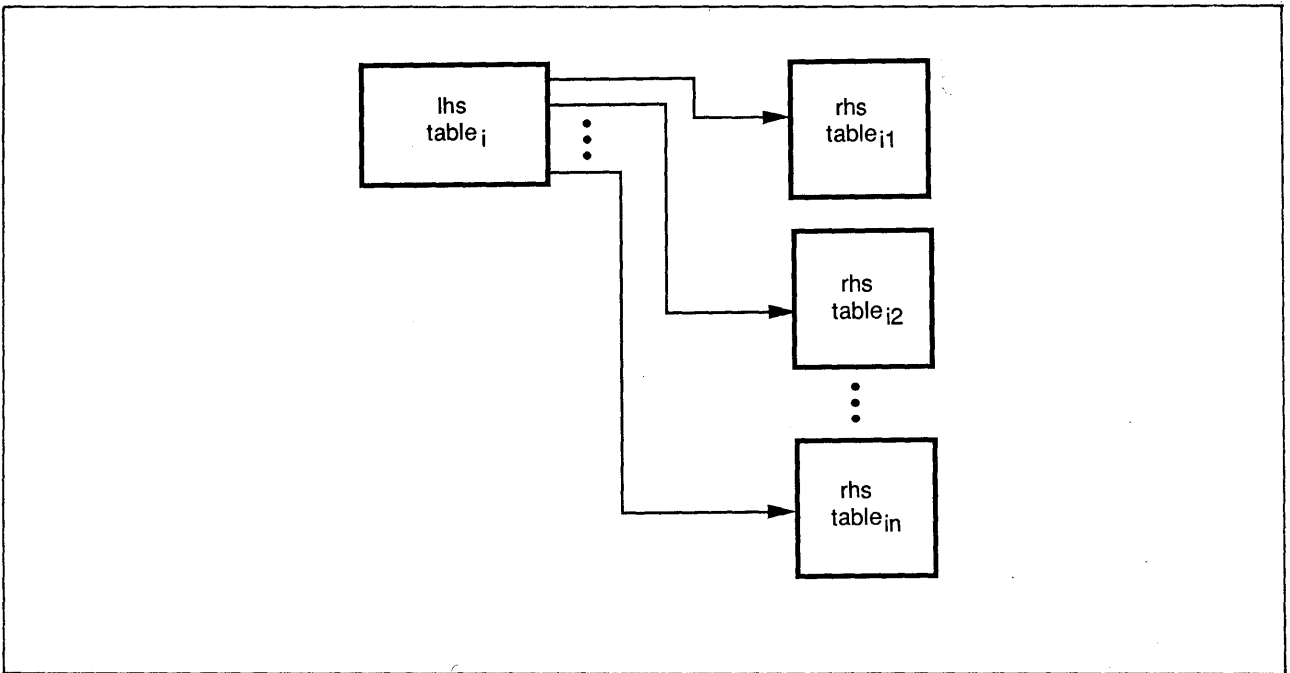


Figure 10-6. lhs Table Pointer Configuration

lhs Table

An lhs table consists of contiguous, variable-length, full-word-aligned entries describing valid keyword expressions. The entries describe the left-hand sides of expressions and, in turn, point to tables whose entries describe valid right-hand sides. A header relates positional and existence requirements of the keywords described by this table.

The lhs table format is shown in figure 10-7. The table header contains two words in the format shown in figure 10-8. Each lhs entry has the format shown in figure 10-9.

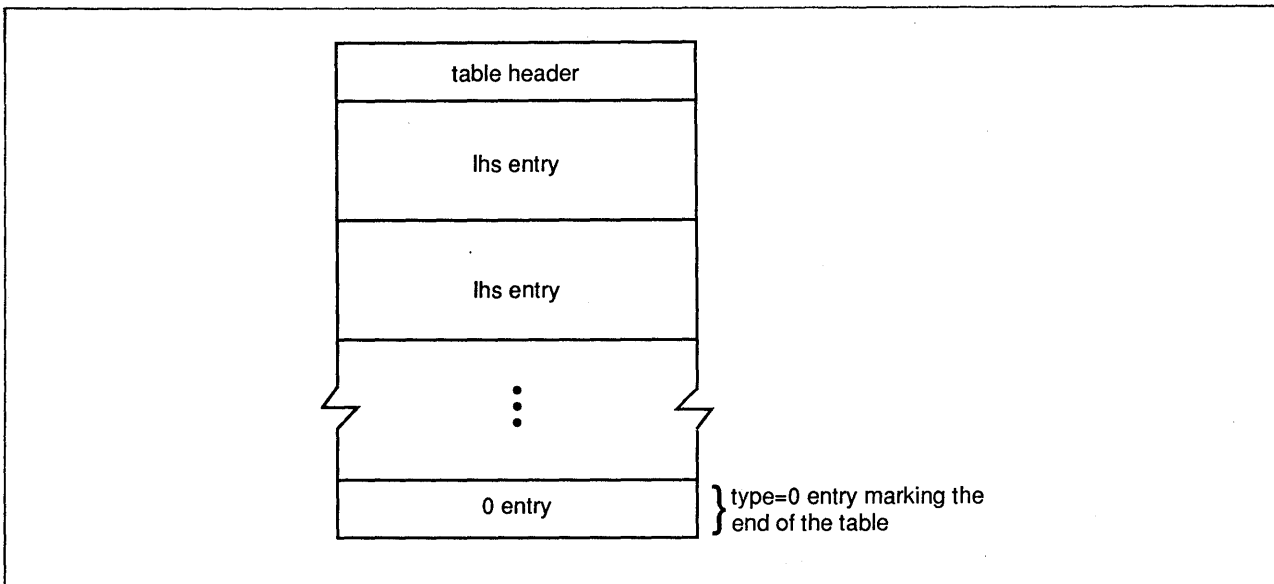
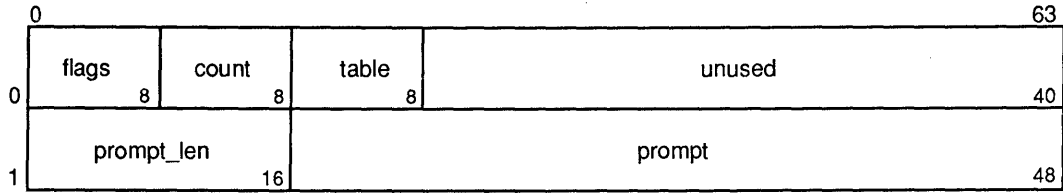
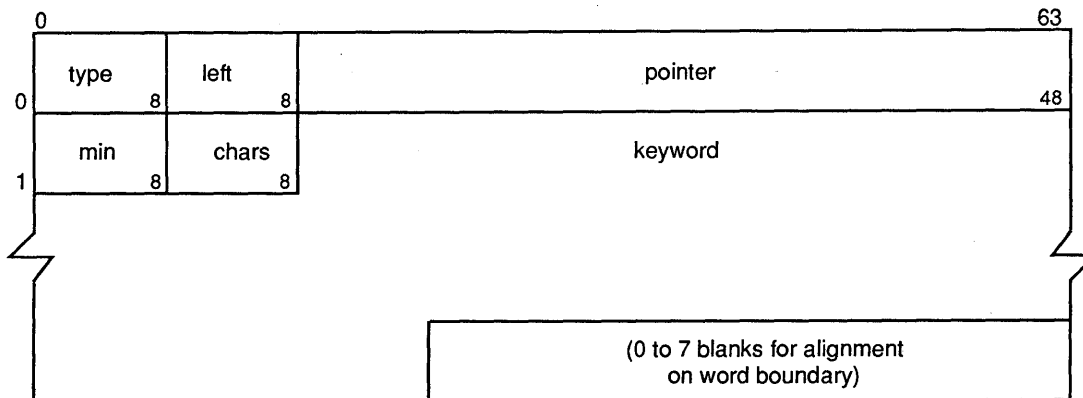


Figure 10-7. lhs Table Format



<u>Word</u>	<u>Field</u>	<u>Description</u>
0	flags	Bits that are set to describe keywords:
	<u>Bit</u>	<u>Description</u>
	6	If 0, entries describe a keyword that is not positional (that is, the keyword described can appear in any order with preceding or succeeding nonpositional keywords); if set to 1, entries describe a positional keyword.
	7	If 0, entries describe an optional keyword; if set to 1, entries describe a required keyword (if no match is found, an error message is issued).
	count	A value that specifies the maximum number of times this table can be used to effect a keyword match.
	table	A value set by the caller and returned in a return buffer entry on a successful lhs and rhs match. (The return buffer is described later in this chapter.)
1	prompt_len	A value that specifies the number of characters in a message whose address is given in the prompt field; valid only when the options bit 60 is set to 1.
	prompt	Address of the text to be output as a prompt to request keywords associated with this table; valid only when the options bit 60 is set to 1. Any ASCII carriage control characters desired must be embedded in the text of the prompting message.

Figure 10-8. lhs Table Header Format



<u>Word</u>	<u>Field</u>	<u>Description</u>
0	type	Entry type. The values are: 0 End of the table. 1 There are no left-hand sides. 2 The keyword expression contains a left-hand side.
NOTE		
	left	A value set by the caller and returned in the return buffer entry upon a successful left-hand side and right-hand side match.
	pointer	Address of the table describing right-hand sides that are valid with this particular left-hand side.
1	min	Minimum number of characters needed in the left-hand side before attempting a substring match against the keyword (type=2).
	chars	The number of characters in the keyword if the type field is 2; must be 0 if the type is 1.
	keyword	Text to be used in validating the left-hand side of the expression (type=2).

Figure 10-9. lhs Table Entry Format

rhs Table

The rhs table contains contiguous, variable-length, full-word-aligned entries that describe valid right-hand side expressions. The table format is shown in figure 10-10. The first word of each rhs entry has the format shown in figure 10-11.

When the type field is 0, the rhs table entry is one word having the format shown in figure 10-11, but with the right, flags, and count fields unused. When the type field is 1, the rhs table entry is one word having the format shown in figure 10-11, but with the flags and count fields unused. When the type field is 2, the format of the rhs table entry is as shown in figure 10-12. When the type field is 3, the format of the rhs table entry is as shown in figure 10-13.

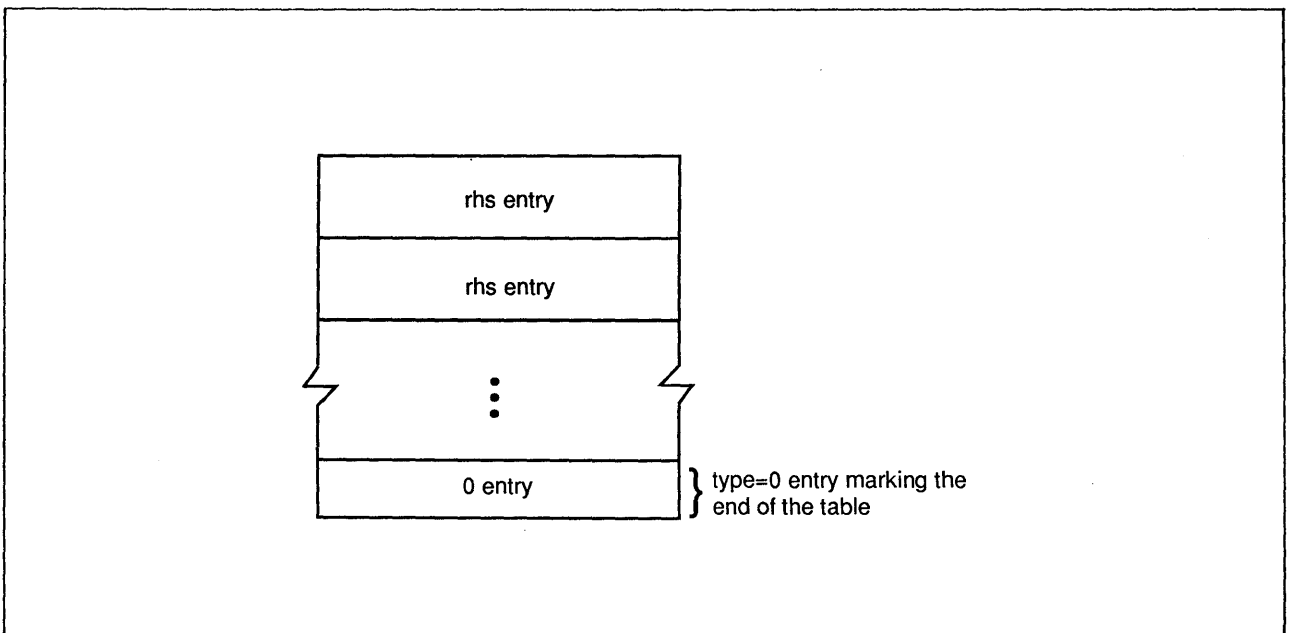
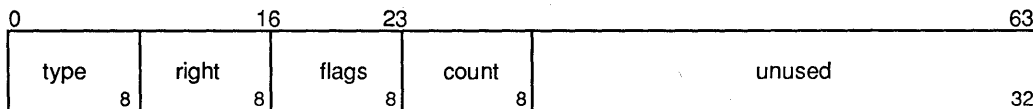


Figure 10-10. rhs Table Format



Field

Description

type Entry type number. The format of each entry and the meaning of its flags and count fields vary according to the entry type. The types are:

- 0 End of the table.
- 1 No right-hand side in the expression.
- 2 Literal; the right-hand side of the expression must match the initial substring of the literal.
- 3 Element list.
- 4 Arbitrary character string is returned in the return buffer.
- 5 All remaining text (255 characters maximum) is returned in the return buffer.
- 6 Numbers in the range of -247-1 to 247-1 are returned in the return buffer.
- 7 File names are returned in the return buffer.
- 8 User number.
- 9 Ignore the keyword.

NOTE

Where both a literal character string and an arbitrary character string may be used as parameters, the arbitrary character string must follow the literal character string for the parameters to be interpreted correctly.

right A value set by the caller and returned in the return buffer entry on a successful left-hand side and right-hand side match.

flags Flag bits, which are set to describe valid right-hand sides of expressions. Bit meanings depend on entry type.

count Maximum number of elements in the right-hand side for those entry types that allow lists.

Figure 10-11. rhs Table Entry Format (First Word)

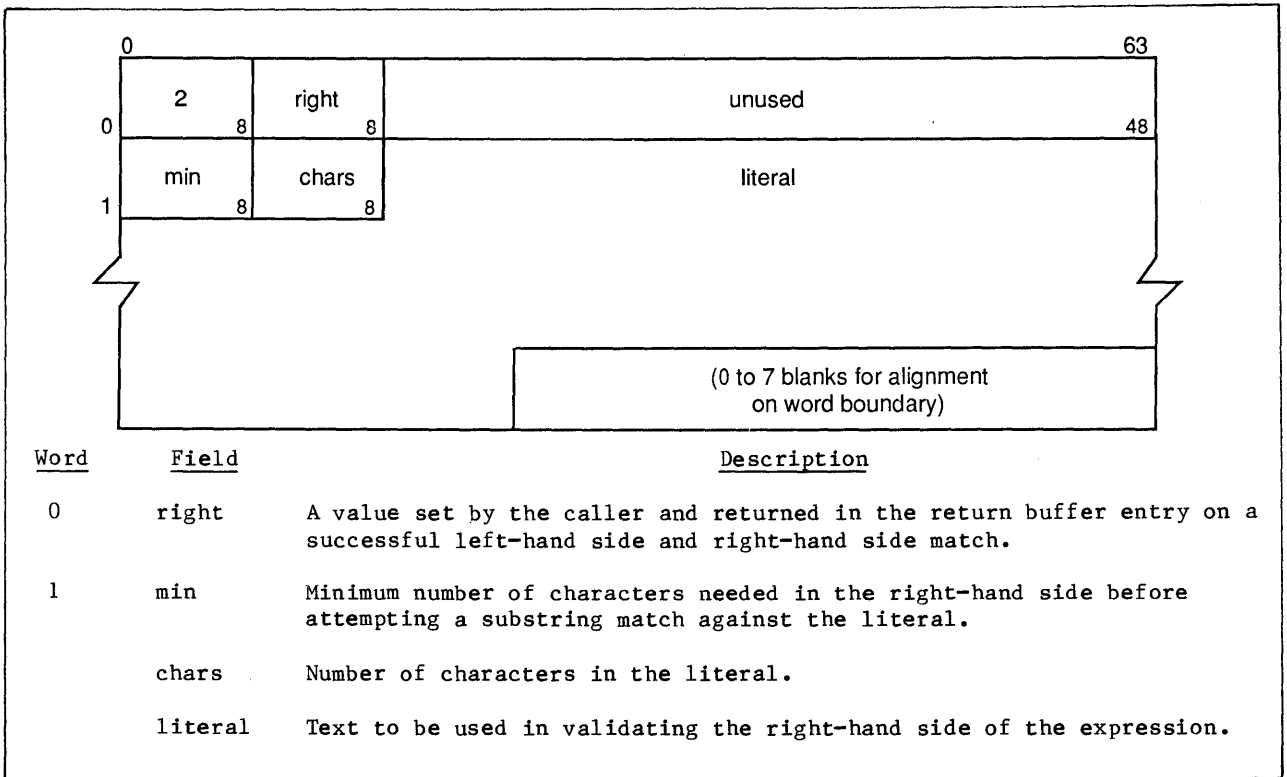


Figure 10-12. rhs Table Entry Format, Type 2

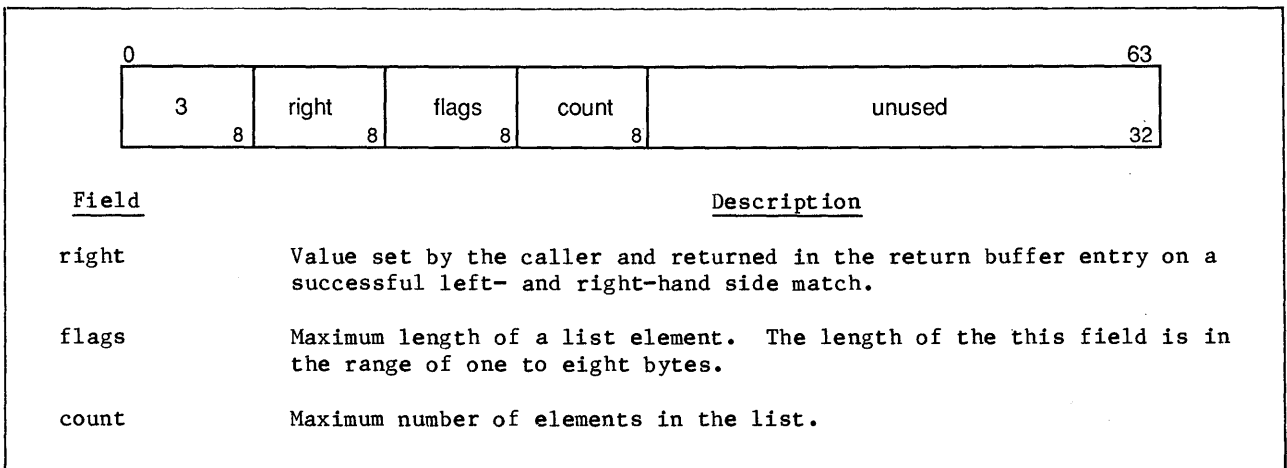


Figure 10-13. rhs Table Entry Format, Type 3

When the type field is 4, the format of the rhs table entry is as shown in figure 10-14, except that the flags field is not used. The right-hand side of the expression contains 1 or more literal character strings (255 characters maximum per literal string are returned in the return buffer). Quotes may be embedded within the literal string by using the double quotation mark character to indicate the presence of a quote. During processing, the string will be appropriately edited. Enclosing quotes are required only if special characters defined in table 10-1 are part of the text.

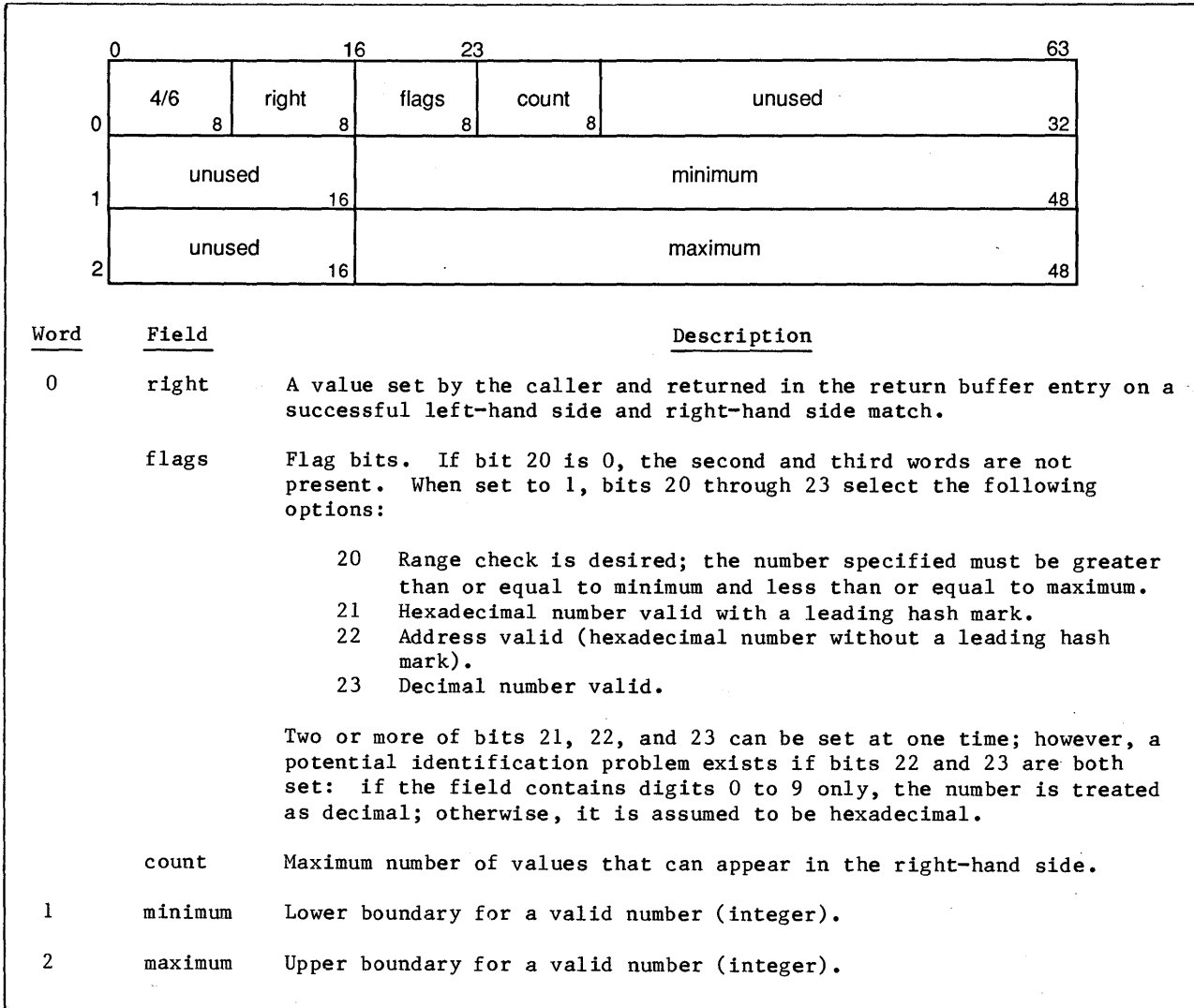


Figure 10-14. rhs Table Entry Format, Type 4/6

When the type field is 5, the format of the rhs table entry is as shown in figure 10-11, except that the flags and count fields are not used. When the type field is 6, the format of the rhs table entry is as shown in figure 10-14.

When the type field is 7, the format of the rhs table entry is as shown in figure 10-11. Four of the individual bits in the flags field can be set to 1, in which case they have the following meanings:

<u>Bit</u>	<u>Description</u>
20	Punch attribute (PU) is valid.
21	Print attribute (PR) is valid.
22	Length attribute can be specified.
23	Drop file name can be specified.

Two or more of the bits can be set at one time. The count field contains the number of file names that appear in the associated return buffer entry.

When the type field is 8, the format of the rhs table entry is as shown in figure 10-11. Two of the individual bits in the flags field can be set to 1, in which case they have the following meanings:

<u>Bit</u>	<u>Description</u>
22	Return an ASCII value.
23	Return a binary value.

One or both of the bits can be set at one time.

When the type field is 9, the format of the rhs table entry is as shown in figure 10-11, except that the right, flags, and count fields are not used. For this type, no entry is made in the return buffer and processing continues with the next keyword expression.

When the rhs table entry types are 1 and 2, the format of the return buffer entry is as shown in figure 10-17. The words field is always 1.

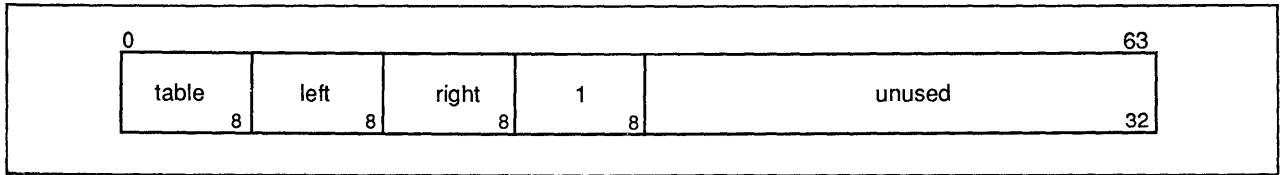
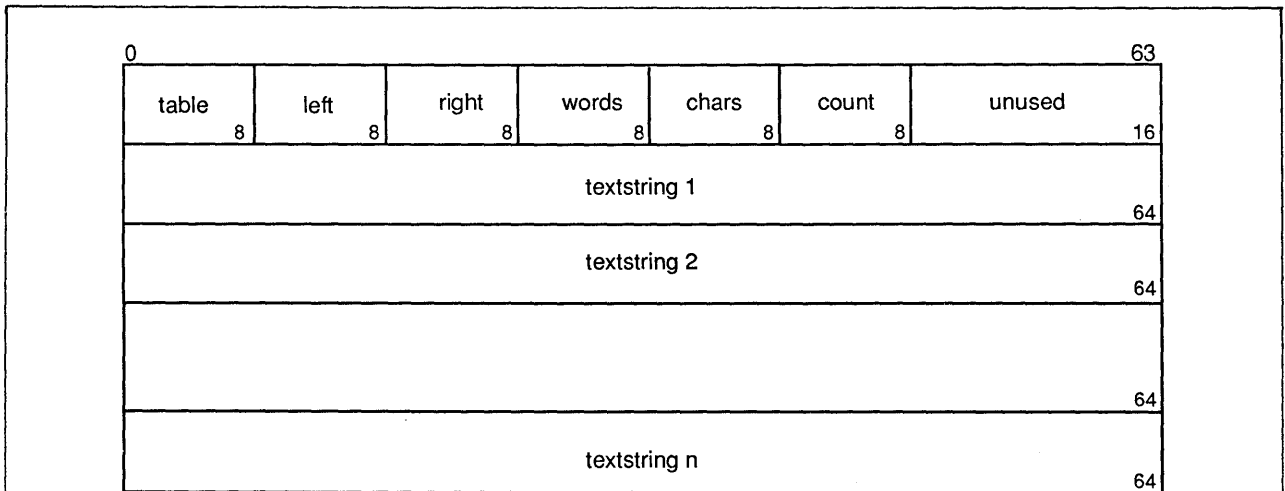


Figure 10-17. Return Buffer Entry Format, Types 1 and 2

When the rhs table entry type is 3, the flags field is used to specify the maximum allowable length of a list element. The allowable range of values is 1 through 8. The format of the return buffer is shown in figure 10-18.



<u>Field</u>	<u>Description</u>
table	Value from the header of the lhs table (t_1) that provided the left- and right-hand side entries affecting the keyword match.
left, right	Values from the left field of the particular lhs table entry and the right field of the rhs table entry affecting the left- and right-hand side matches.
words	Total number of words in this particular entry.
chars	The meaning of this field varies with the type of the right-hand side. It is the length of the returned information or the length of an element of a returned list, such as list of file names.
count	Number of elements returned.
text string i	Returned elements, ASCII left-justified and blank-filled.

Figure 10-18. Return Buffer Entry Format, Type 3

When the rhs table entry type is 4, the format of the return buffer entry is shown in figure 10-19. Since the multiple literal strings will likely be variable in length, the format of the return buffer returned for type 4 differs from the format of all other return buffers. A header word will precede each literal string returned.

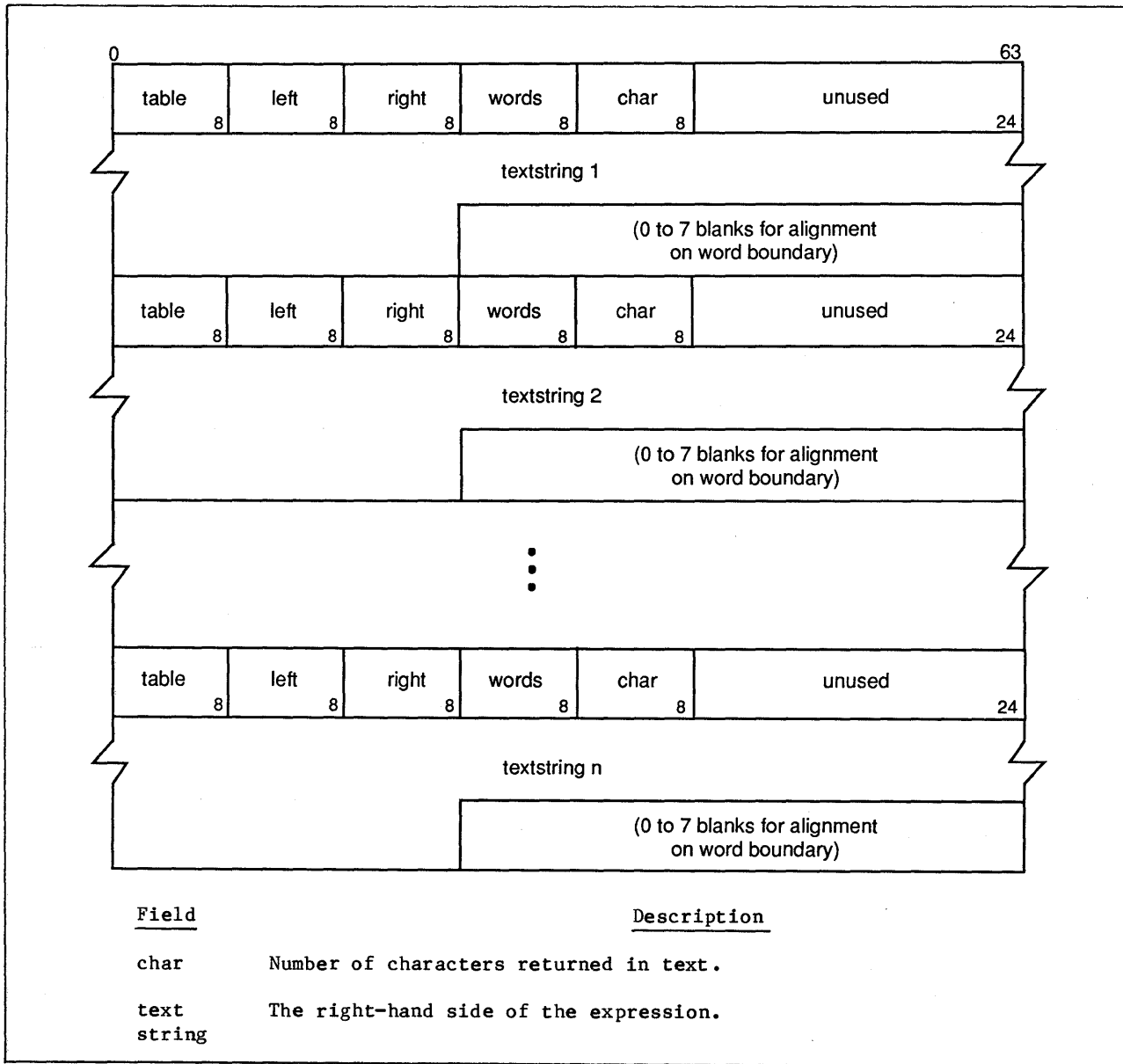


Figure 10-19. Return Buffer Entry Format, Type 4

When the rhs table entry type is 7 with flag bits 20, 21, and 22 all set to 0, the format of the return buffer entry is as shown in figure 10-22. The chars field is always 8.

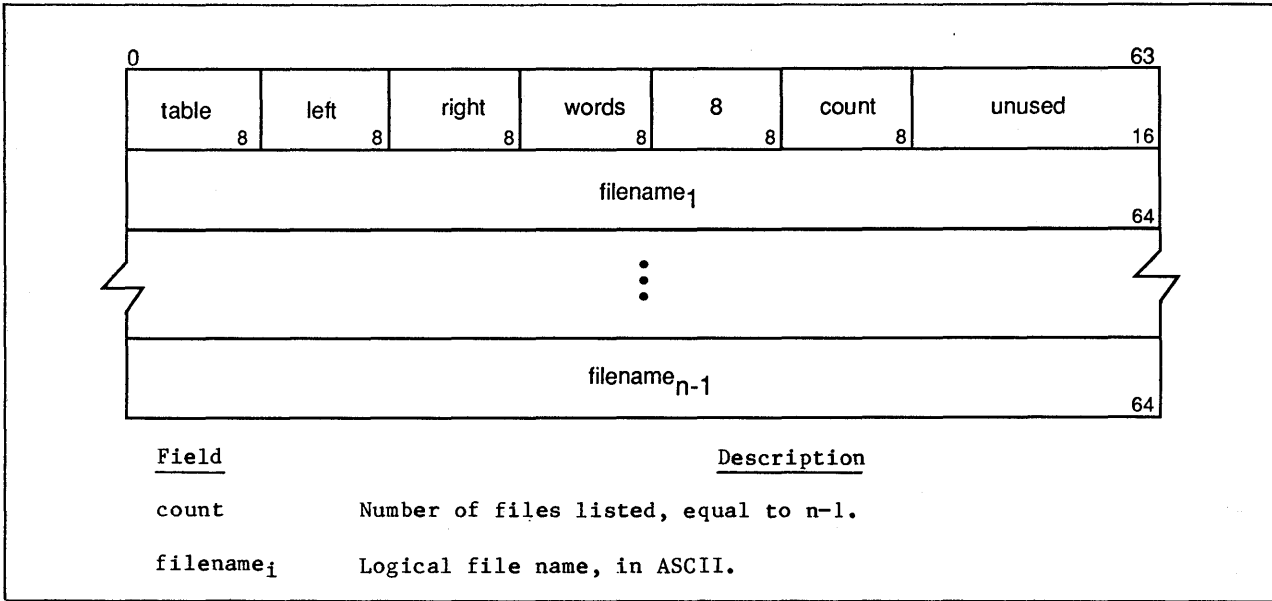


Figure 10-22. Return Buffer Entry Format, Type 7 with Zeroed Flags

When the rhs table entry type is 7 with flag bits 20, 21, or 22 set to 1, the format of the return buffer entry is as shown in figure 10-23. The chars field is always 16.

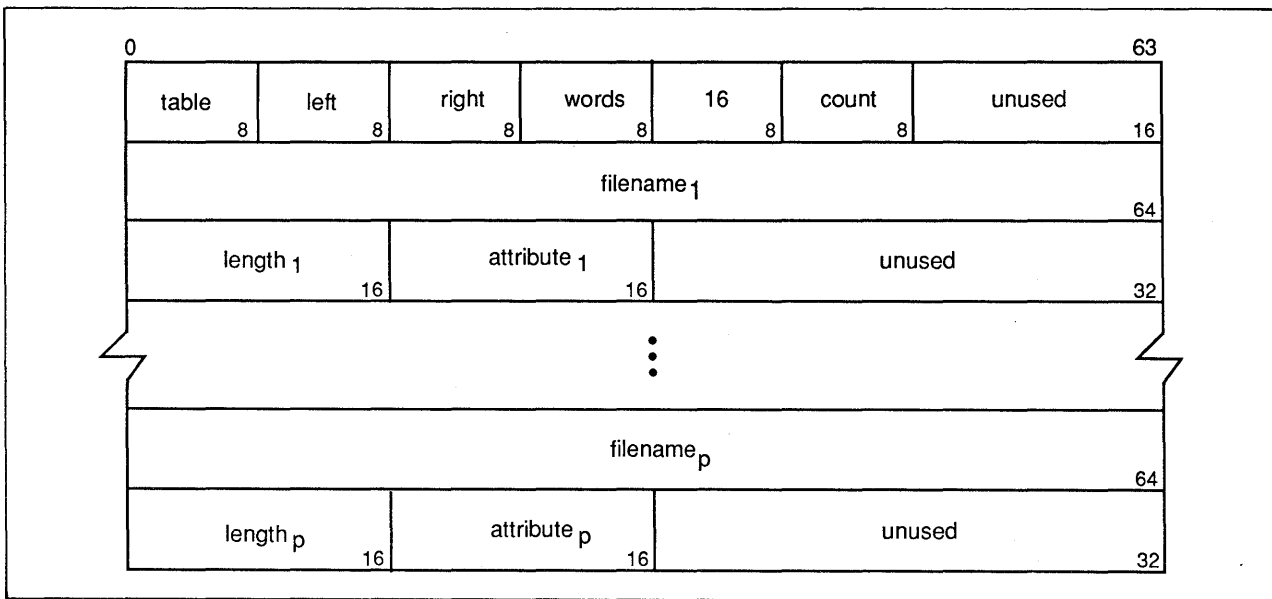


Figure 10-23. Return Buffer Entry Format, Type 7 with Set Flags (Sheet 1 of 2)

<u>Field</u>	<u>Description</u>
count	Number of files listed; has the value (n-1)/2.
filename _i	Name of the file specified, left-justified with blank fill.
length _i	Length of the file in small pages. If not specified, binary 0 is returned.
attribute _i	File attribute: ASCII punch (PU) or print (PR). If not specified, blanks are returned.

Figure 10-23. Return Buffer Entry Format, Type 7 with Set Flags (Sheet 2 of 2)

When the rhs table entry type is 8 with only one of flag bits 22 and 23 set, the format of the return buffer entry is as shown in figure 10-24. The chars field is always 8.

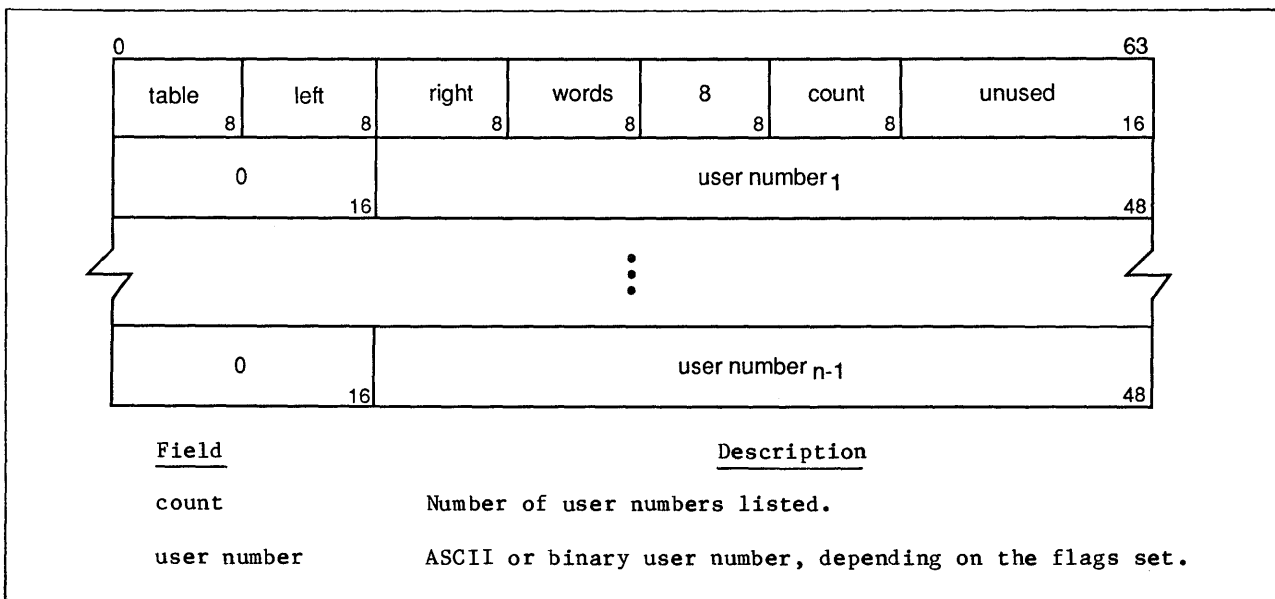


Figure 10-24. Return Buffer Entry Format, Type 8 with One Set Flag

Special Characters

The Q7KEYWRD subroutine scans for special characters in the execute line text to extract keyword expressions. These characters and their meanings (under given conditions) are described in table 10-1.

Table 10-1. Execute Line Special Characters

Character	Description
"	Delimits a literal character string on the right-hand side of the expression. An embedded quote within a literal character string must be represented by the double quotation mark character; for example, "AB""C""DE" would be the representation of the literal string AB"C"DE. Each string of the right-hand side must be enclosed in quotes if it includes a special character of table 10-1. Q7KEYWRD will not perform concatenation of a literal in quotes and other character strings.
blank	Delimits a keyword expression unless it occurs within a literal character string.
,	Delimits a keyword expression unless it occurs within a literal character string.
=	Separates the left- and right-hand sides of keyword expressions unless they occur within a literal character string.



This chapter contains information about the VSOS screen utilities, Q9SCR and Q9SPRINT, for use in generating screen format displays. This information is for a system programmer who is interested in developing an application or utility using screen formatting displays. The Q9SPRINT utility has usefulness apart from generating screen formatted displays in that it makes it easier for an application, especially a non-FTN200-application, to create formatted text strings. This chapter also contains a brief set of definitions and guidelines for implementing screen displays to be managed by the VSOS display manager and to be viewed by the DTA and DTN display commands.

Q9SCR

Q9SCR is a routine which implements a subset of the Unix† curses library to make it easier for an application to create and manipulate one or more dynamic windows for the display of information, usually on a terminal screen. These displays are used in various ways: some can be created in response to a specific user input. Other applications may create dynamic displays as their main method of interaction with the user. Still other applications may build progress displays always, even though those displays are looked at only seldom (for example, when the program seems to be having trouble).

Displays created by Q9SCR can be made visible to the user in three different ways: if the user has access to an MCU operator's console, the displays can be viewed with the DTA or DTN commands. If the user is logged on through ITFS (or directly at an MCU terminal), Q9SCR can generate the control and escape sequences used by many smart terminals. These strings are then sent by the application by way of the normal Q5SNDMCR or Q5SNDMJC calls to the user's terminal. Lastly, the application can use Q9SCR to build up a full screen, piecemeal, and then dump it all at once as normal SIL R-format text, either to a dumb terminal or to a file for printing. All three of these capabilities are available at the same time, so that, for example, a program could drive a display on a user's terminal which was also available to the MCU operator and which could be preserved on a file by way of a snapshot command.

Q9SCR makes no SIL calls itself, and neither does the only routine it calls (Q9SPRINT). Thus, it is suitable for use by VSOS resident or virtual.

Appendix G is a sample display program that illustrates the use of many of the functions of Q9SCR.

†Unix is a trademark of AT&T Bell Laboratories.

DEFINITIONS

These words have specific meanings in relation to Q9SCR: screen, window, and subwindow. A screen is a container for windows. Most often, a screen corresponds to the whole viewing area of a terminal (for example, 24 lines of 80 columns each), although it may represent a sheet of printer output. Screens may be declared larger or smaller than the physical device they describe, but this is likely to be less useful than a screen which exactly matches the device. A user application will normally describe only one screen, since most applications interact with only one terminal at a time. VSOS resident and virtual can use three screens, one for each MCU terminal.

A window is a rectangular region, which, if displayed, will affect all or a portion of a screen. There may be several, possible overlapping, windows associated with a screen at any given time. The contents of the window are independent, and changes to one window do not affect another window. Whenever two or more windows overlap on the screen, the information from the window refreshed most recently will determine the screen's contents in the region of overlap.

A subwindow is a rectangular region wholly contained in another window and which shares the same contents as the containing (parent) window in the region of overlap. Subwindows make it easy to limit modifications to a portion of a window.

SYNOPSIS

The supported functions and their arguments are summarized below:

```
RESULT = Q9SCR( ADDCH, CHAR )
RESULT = Q9SCR( ADDSTR, STRING, LEN )
RESULT = Q9SCR( BOX, VERT-CH, HORIZ-CH, ROWS, COLS, TOP, LEFT )
RESULT = Q9SCR( DLEAR )
RESULT = Q9SCR( CLRTOBOT )
RESULT = Q9SCR( CLRTOEOL )
RESULT = Q9SCR( DELCH )
RESULT = Q9SCR( DELETELN )
RESULT = Q9SCR( DELWIN )
RESULT = Q9SCR( ENDSCR, SCREEN-NUMBER )
RESULT = Q9SCR( ERASE )
RESULT = Q9SCR( GETCOL )
WPTR   = Q9SCR( GETCUR )
ADDR   = Q9SCR( GETMCU, SCREEN-NUMBER )
RESULT = Q9SCR( GETROW )
WPTR   = Q9SCR( GETSTD, SCREEN-NUMBER )
RESULT = Q9SCR( INCH )
WPTR   = Q9SCR( INITSCR, BUFFER, LEN, NSCREENS, OPTIONS, ROWS, COLS )†
WPTR   = Q9SCR( INITSCR, OPTIONS )†
RESULT = Q9SCR( INSCH, CHAR )
RESULT = Q9SCR( INSERTLN )
RESULT = Q9SCR( LOOKUP, ITEM )
MEMPTR = Q9SCR( MALLOC, LENGTH, FLAG )
RESULT = Q9SCR( MFREE, MEMPTR )
RESULT = Q9SCR( MVWIN, NEWROW, NEWCOL )
WPTR   = Q9SCR( NEWIN, ROWS, COLS, S-ROW, S-COL, OPTIONS, SCREEN )†
RESULT = Q9SCR( NULL )
RESULT = Q9SCR( OVERLAY, SOURCE-WINDOW )
RESULT = Q9SCR( OVERWRITE, SOURCE-WINDOW )
RESULT = Q9SCR( PRINTW, FORMAT, VALUE, ... )
LEN    = Q9SCR( REFRESH, RETURN-BUFFER, TERMINAL-TYPE )
WPTR   = Q9SCR( RESET, SCREEN-NUMBER )
RESULT = Q9SCR( SCROLL )
RESULT = Q9SCR( SCROLLOK, FLAG )
WPTR   = Q9SCR( SUBWIN, ROWS, COLS, FIRST-ROW, FIRST-COL )
```

Underlined arguments are required; the others are optional. See the next section on calling conventions for how the presence of these arguments is indicated.

†These functions do not accept the optional arguments WRTR, NEWROW, and NEWCOL.

CALLING CONVENTIONS

The general form of a call to Q9SCR is:

```
RESULT = Q9SCR( FUNCTION-SELECTOR + MODIFIERS, ARGUMENTS ... )
```

where FUNCTION-SELECTOR is a small integer which selects which of Q9SCR's functions is desired, and ARGUMENTS are the parameters appropriate to that function. The synopsis above indicates that most functions have optional arguments of WPTR and NEWROW, NEWCOL. If the WPTR argument is given, it immediately follows the FUNCTION-SELECTOR and the FUNCTION-SELECTOR has the MODIFIER WIND added to it. If NEWROW and NEWCOL are given, they come next, and their presence is indicated by increasing MODIFIERS by MOVE. For example, the basic call to place a character (for example, A) at the current row and column on the current window is:

```
RESULT = Q9SCR( ADDCH, 'A' )
```

to move to a different location before putting the character, the call is:

```
RESULT = Q9SCR( ADDCH + MOVE, NEWROW, NEWCOL, 'A' )
```

to put the character at the current row and column of an alternate window is:

```
RESULT = Q9SCR( ADDCH + WIND, WPTR, 'A' )
```

and to both change windows and move before putting the character is:

```
RESULT = Q9SCR( ADDCH + WIND + MOVE, WPTR, NEWROW, NEWCOL, 'A' )
```

The default window (called the current window) is the window used by the previous Q9SCR call. However, if the WPTR argument is present, then that window becomes the current window.

Q9SCR is a function, rather than a procedure. Its typical return value is just a success/failure flag. Success is normally indicated by a non-zero value in the low 48 bits. A failed call returns a floating-point zero, with an error indicator encoded in the exponent and the low 48 bits zero. Thus, FTN200 programs should declare Q9SCR as a real-valued function and should assign its result to a real variable. IMPL and C programs can either do this or treat Q9SCR as returning a structured variable.

See table 11-1 for the possible error results. In general, any errors other than ENOMEN or ESCROLL indicate a serious programming error. ENOMEN is usually also serious, unless the application is one which allows the user to create multiple windows dynamically. In this case, the application can watch for ENOMEN to indicate that the maximum number of windows has been reached and the current user request cannot be satisfied.

Since ESCROLL just indicates that the application tried to put more data in a window than the window could hold, this error is not serious. The programmer may want to enlarge the window or shrink the data. Alternatively, if the data are of a sequential nature, perhaps the window should be made into a scrolling window with the SCROLLOK function.

As a general rule, once a program is debugged, it is not necessary to check the return result from each Q9SCR call. However, the result from DELWIN, ENDSCR, INITSCR, MALLOC, MFREE, NEWWIN, and SUBWIN should always be checked for the EBOTCH error, since this means Q9SCR's memory pointers have been damaged and further calls on Q9SCR (except for another INITSCR call) could cause Q9SCR to alter unpredictable areas in the application's memory.

Table 11-1. Symbols Known by LOOKUP

In addition to the names of all the function selectors supported by Q9SCR, the lookup function will return values for the following symbols:

Error Numbers	
EARGC	The function call includes the wrong number of arguments, either too many or too few. This often means the WPTR or NEWROW, NEWCOL arguments were supplied, but the WIND and MOVE bits were not added to the FUNCTION-SELECTOR.
EBOTCH	Q9SCR's memory allocation information is damaged, possibly because the INITSCR buffer was overwritten. It is unsafe to make any further calls to Q9SCR.
ECOL	The NEWCOL argument is negative or too big for the window.
EINIT	INITSCR must successfully complete before any functions other than LOOKUP can be called.
ENOCURR	There is no current, default window (for example, after a DELWIN or ENDSCR call or after an EWPTR error) and the current call does not include the WIND option.
ENOMEM	The buffer supplied to INITSCR is too small.
EROW	The NEWROW ARGUMENT IS NEGATIVE OR TOO BIG FOR THE WINDOW.
ESROLL	A non-scrolling window would have to scroll to satisfy the call to Q9SCR.
EQPTR	The WPTR argument is not a valid window pointer.
EFUNC	The FUNCTION-SELECTOR is unknown.
EARG 1 ... EARG 6	The n'th argument's value is illegal.
Function selector modifiers	
WIND	The first argument is a pointer to a window to be made the current window.
MOVE	The first two arguments, after any WIND argument, are new values for the row and column positions of the cursor.
Options	
OPDIR	Modifications to the window should go directly to the screen buffer.

Terminal Type Names	
ANSI	Terminal using the ANSI escape sequences
CDC628	The normal terminal supplied with the 205 MCU
CDC752	Another 205 MCU terminal (same as CDC628 to Q9SCR)
I100	Infoton 100
VT52	DEC VT-52
VT100	DEC VT-100 (same as ANSI so far as Q9SCR is concerned)
z29	Zenith-29, Healthkit-29

The GETCOL, GETROW, and INCH calls might succeed, but still return zero. Since these calls are unlikely to fail, it is easiest for an FTN200 program to just accept the result as correct, rather than try to distinguish between a successful, integer zero and a failed, floating point zero.

Some Q9SCR functions return window pointers (WPTR). These pointers are not to be used by the callers in any way except as arguments to other Q9SCR calls. That is, no arithmetic of any kind should be performed on them, including conversion from floating point to integer.

The values for FUNCTION-SELECTOR, WIND, MOVE, and the various error codes are defined in the Q9SCRCOM common deck (where all names have a prefix of SCR_). Since these definitions are useful only to IMPL routines, an alternate method is provided for FTN200 and C (and IMPL) routines to determine the codes at runtime. The call

```
VALUE = Q9SCR (1, 'ADDCH' )
```

where NAME is the name of one of the symbols in the Q9SCRCOM deck, returns the value for that symbol. Continuing the example from above, one can get the values for ADDCH, WIND, and MOVE by:

```
ADDCH = Q9SCR( 1, 'ADDCH' )
WIND  = Q9SCR( 1, 'WIND'  )
MOVE  = Q9SCR( 1, 'MOVE'  )
```

Normally, one would look up all the needed values only once in an initialization routine and save them in a common block for later use by other routines.

By convention, screens are enumerated starting from one, but row and column positions are zero based.

MEMORY USE

Q9SCR needs some memory to maintain tables and buffers. Because Q9SCR must be usable by the VSOS resident kernel, it cannot allocate this memory through the normal SIL Q5MEMORY mechanism. Instead, the first call to Q9SCR (except for the name lookup calls mentioned above) must be for the INITSCR function and should specify a buffer allocated by the caller for use by Q9SCR. Q9SCR remembers this buffer's location for future calls until the next INITSCR call, if any. The calling program should not modify the buffer in any way between the INITSCR call and the last use of Q9SCR.

See the description of the INITSCR call for some guidelines for memory needs.

DESCRIPTIONS OF INDIVIDUAL ROUTINES

Each routine is described following a copy of its usage synopsis. The same conventions are used as before: mandatory arguments are underlined routines which don't accept the WINDOW and NEWROW, NEWCOL arguments marked with a footnote.

RESULT = Q9SCR(ADDCH, CHAR)

CHAR is a single character, either right-justified in a word or the first character of a FTN200 character variable or literal. It replaces the character at the current row and column on the window and the cursor is advanced by one position. If the cursor advances past the end of a line, it is positioned to the first column of the next line. If it was already on the last line in the window and scrolling is permitted, the window is scrolled up one line, with the previous top line being discarded. If scrolling is disabled, the ESCROLL error is returned and the cursor stays at the end of the window.

ASCII control characters are handled slightly differently: line-feed and unit-separator (VSOS end-of-line, #1F) cause the remainder of the current line to be set to spaces and the cursor advanced to the next line, with possible scrolling as above. Carriage-return causes the cursor to move to the first character of the current line. Horizontal tab (#09) is replaced by spaces to the next column position which is a multiple of 8. All other control characters are replaced by the two character sequence c, where c is the original character with the #40 bit toggled.

RESULT = Q9SCR(ADDSTR, STRING, LEN)

STRING is a string of characters to be placed into the window as if by repeated calls to the ADDCH function. If LEN is present, it gives the numbers of characters to use from string. If LEN is absent and STRING is a FTN200 character variable or literal, then its natural length is used. Otherwise, the first character of STRING is assumed to be a delimiter character and all characters after this delimiter, but before the next occurrence of the delimiter in the string, are used as the text.

If adding a character from STRING causes an ESCROLL error, the rest of STRING is not processed.

RESULT = Q9SCR(BOX, VERT-CH, HORIZ-CH, ROWS, COLS, TOP, LEFT)

The BOX function draws a rectangle within a window. VERT-CH is the character used for the sides of the box and HORIZ-CH is used for the top and bottom. Both characters should be either FTN200 character variables or literals or should be right-justified in a word. TOP and LEFT give the row and column, respectively, of the top, left corner of the box. ROWS and COLS give the height and width of the box. Either all four parameters ROWS, COLS, TOP, and LEFT must be given, or all four must be omitted. If they are omitted, the box is drawn as large as possible within the window. This is useful to frame a window. If ROWS or COLS equals 1, then the box becomes a line. If the box includes the bottom, right corner of the screen, then the corners of the box are set to spaces, to prevent scrolling.

This box is part of the window and is treated no differently from other text. To protect a box used as a frame, one can create a subwindow just inside the main window and do all writing on the subwindow.

RESULT = Q9SCR(CLEAR)

CLEAR sets the whole window to spaces. If the window covers the whole screen, then the next time the window is REFRESHED, Q9SCR includes the appropriate escape sequence to clear the terminal's screen before updating the display.

RESULT = Q9SCR(CLRTOBOT)

CLRTOBOT sets the window to spaces from the current cursor position to the last character of the last row.

RESULT = Q9SCR(CLRTOEOL)

CLRTOEOL sets the remainder of the current line to spaces.

RESULT = Q9SCR(DELCH)

The DELCH function deletes the character under the cursor and moves the remaining characters on the same line left one space. The last character of the line is set to space.

RESULT = Q9SCR(DELETELN)

The DELETELN function deletes the entire line the cursor is in. The remaining lines in the window move up one line and the last line is set to spaces.

RESULT = Q9SCR(DELWIN)

A window is destroyed and its storage made available to Q9SCR for other uses with the DELWIN function. If the window has subwindows, the subwindows and any of their sub-subwindows, and so forth, are also destroyed.

Note that deleting a window which covers another window does not automatically cause the contents of the now-uncovered window to appear on the screen. The uncovered window should be REFRESHED if this is desired.

After a window is deleted, Q9SCR's notion of the current window becomes undefined, so the next call to Q9SCR should explicitly select a window.

RESULT = Q9SCR(ENDSCR, SCREEN-NUMBER)

ENDSCR causes all windows associated with the screen specified by SCREEN-NUMBER (default is the screen containing the current window) to be destroyed and all memory associated with the screen to be put back in Q9SCR's available pool.

RESULT = Q9SCR(ERASE)

The ERASE function is identical to the CLEAR function, except that it does not cause an explicit terminal screen clear on the next REFRESH.

RESULT = Q9SCR(GETCOL)

This function returns the column position of the cursor in the current window. Note that the return value is zero if the cursor is at the left edge of the window, so this function is an exception to the rule that success is indicated by a non-zero return value.

WPTR = Q9SCR(GETCUR)

The application can get a pointer to the current window with the GETCUR function. This might be used in a subroutine to save the current window selection across modifications to an alternate window. The original window could be restored as the default window by way of the NULL function with the WIND modifier.

ADDR = Q9SCR(GETMCU, SCREEN-NUMBER)

The GETMCU function returns the bit address of the MCU buffer associated with the specified screen. This address can then be used in a VSOS #001E system message to register the screen with the Display Task Manager. If SCREEN-NUMBER is omitted, the screen containing the current window is used.

RESULT = Q9SCR(GETROW)

This function returns the row position of the cursor in the current window. Note that, like GETCOL, the return value can be zero (if the cursor is in the top line of the window).

WPTR = Q9SCR(GETSTD, SCREEN-NUMBER)

The GETSTD function returns a window pointer which refers to the standard window for the specified screen. If SCREEN-NUMBER is omitted, then the screen containing the current window is used.

The standard window is normally created by the INITSCR call and is the same size as the screen(s). However, the standard window for a screen can change or become undefined as a result of DELWIN function calls.

RESULT = Q9SCR(INCH)

This function returns the character under the cursor, right-justified, zero filled. Note that the result can be zero (if the character is a null.)

WPTR = Q9SCR(INITSCR, BUFFER, LEN, NSCREENS, OPTIONS, ROWS, COLS)†
WPTR = Q9SCR(INITSCR, OPTIONS)

The INITSCR function initializes Q9SCR for all later calls. It specifies the region of memory available to Q9SCR as starting at BUFFER and extending for LEN words. The default buffer is #800 words starting at address #60000.

The NSCREENS parameter gives the number of screens Q9SCR is to manage (default is 1). Each screen consists of ROWS lines of COLS columns (default is 24 rows by 80 columns). The INITSCR call does an implicit NEWWIN for a window the same size as the screen. This window becomes the standard window for the screen.

The OPTIONS parameter is normally zero, but can be OPDIR if the standard window is to be treated more like a subwindow than a normal window. That is, modifications to the standard window affect the screen buffer directly, without a REFRESH call. This is useful to reduce memory requirements for programs which need only to create displays for the MCU or for printers.

Note that if there is only one parameter to INITSCR, that parameter is assumed to be OPTIONS.

Each screen is allocated a buffer in which Q9SCR keeps track of what it thinks the terminal screen looks like. This same buffer can be displayed by the MCU DTA or DTN commands, although it looks right only if COLS=80 and ROWS is at least 20. Only screen rows 0 through 19 appear on the MCU display, shifted down to allow for the normal three MCU header lines. Row 19 is then right above the command entry line of the MCU terminal.

There is no simple formula to estimate the size needed for the buffer, but a rough approximation is: $512 * (1 + NSCREENS)$ words. Each additional window beyond the default needs $(16 + ROWS)$ words for tables plus $(ROWS * COLS + 7) / 8$ words of text buffers. Each subwindow needs $(16 + ROWS)$ words. MALLOCed memory requires $(4 + LENGTH)$ words.

INITSCR returns a pointer to the standard window for screen 1. The GETSTD function can be used to get the pointers the other screens' standard windows.

†These functions do not accept the optional arguments WRTR, NEWROW, and NEWCOL.

RESULT = Q9SCR(INSCH, CHAR)

INSCH inserts the character CHAR at the current cursor position. The characters to the right of the cursor are shifted right to make room, and the last character on the line is deleted. If the cursor is initially at the bottom, right corner of the window, scrolling is not permitted on the window, and CHAR is not a space, then ESCROLL is returned and the window is not altered.

CHAR should be either right-justified in a word or an FTN200 character variable or literal. If CHAR is not a valid, printable ASCII character, it is replaced by ?.

RESULT = Q9SCR(INSERTLN)

The INSERTLN function inserts a blank line above the current line by moving all lines starting with the current line, down one line (discarding the bottom line) and then filling the current line with spaces. The cursor stays at its current position on the screen and thus ends up in the new, blank line.

RESULT = Q9SCR(LOOKUP, ITEM)†

The LOOKUP function takes ITEM, which is normally a character string representing the name of one of Q9SCR's function selectors or options, and returns the internal value for the name. LOOKUP also accepts error codes generated by Q9SCR and returns a Hollerith (8H...) name for the error, and vice versa. In a similar manner, LOOKUP translates a terminal type name into the small integer needed by the REFRESH function.

The FUNCTION-SELECTOR for LOOKUP is always 1.

See table 11-1 for a list of the symbols known by LOOKUP.

MEMPTR = Q9SCR(MALLOC, LENGTH, FLAG)

If an application has several different displays it might generate in response to user input, and many of those displays need temporary work areas while they are active, then a significant amount of memory could be consumed by the work areas associated with inactive displays. If only one screen can be active at a time, then the application could simply reuse one work area. However, if the application can have more than one screen active at the same time, this technique does not work.

To provide for such applications, Q9SCR supports the MALLOC function, which allocates memory back to the application from the buffer given to Q9SCR on the INITSCR call. The LENGTH argument gives the number of words needed. If the FLAG argument is present and non-zero, then the memory is allocated on a block (512 word) boundary. Otherwise, the memory is on a 4 word boundary.

The MALLOC function returns the word address of the allocated memory (for example, the bit address divided by 64).

Allocated memory is associated with a screen (the screen containing the current window). If the screen is released by way of the ENDSCR or RESET functions, then all of its allocated memory is freed for reuse at the same time. The application can also free memory explicitly with the MFREE function.

†These functions do not accept the optional arguments WRTR, NEWROW, and NEWCOL.

```
RESULT = Q9SCR( MFREE, MEMPTR )
```

Memory allocated to the application with the MALLOC function is returned to Q9SCR's available pool with the MFREE function. The MEMPTR argument is the word address of the memory, as returned by the MALLOC function. The application must be careful not to reference the memory after it has been freed.

```
RESULT = Q9SCR( MVWIN, NEWROW, NEWCOL )
```

A window can be moved to a different location on its screen by way of the MVWIN function. The NEWROW and NEWCOL arguments give the new screen row and column where the upper, left corner of the window should appear. The new position must not cause part of the window to extend past the boundary of the screen.

If the window being moved is a subwindow, then the same description applies, except that NEWROW and NEWCOL are relative to the parent window and the result must leave the subwindow within the parent window.

When a window is moved, all of its subwindows are moved also.

```
WPTR = Q9SCR( NEWWIN, ROWS, COLS, S-ROW, S-COL, OPTIONS, SCREEN )†
```

A new window and its associated data structures are created by way of the NEWWIN function. The S-ROW and S-COL arguments give the screen coordinates of the upper, left corner of the window. The ROWS and COLS arguments give the size of the window, in rows and columns, respectively. If either ROWS or COLS is 0, then the largest value which will fit, based on S-ROW or S-COL and the size of the screen, is used.

The OPTIONS parameter is the sum of selected option values. Currently the only option is OPDIR, which requests that the window be created as a pseudo-subwindow of the screen, rather than use a separate text buffer (see the discussion with the INITSCR function).

The SCREEN argument selects which screen the window will be displayed upon by the REFRESH function.

```
RESULT = Q9SCR( NULL )
```

The NULL function does nothing. Its purpose is to let the application select a different window or move the cursor without changing anything else.

Since the FUNCTION-SELECTOR for null is 0, the following two calls do the same thing:

```
RESULT = A9SCR( NULL + MOVE, NEWROW, NEWCOL )  
RESULT = Q9SCR( MOVE, NEWROW, NEWCOL )
```

```
RESULT = Q9SCR( OVERLAY, SOURCE-WINDOW )
```

The OVERLAY function replaces the contents of the current window with the contents of SOURCE-WINDOW in the region where they overlay on the screen. Spaces in SOURCE-WINDOW are not copied, so the current window retains its values in those positions where SOURCE-WINDOW is blank.

†These functions do not accept the optional arguments WRTR, NEWROW, and NEWCOL.

RESULT = Q9SCR(OVERWRITE, SOURCE-WINDOW)

OVERWRITE is identical to OVERLAY, except that spaces are not treated specially by OVERWRITE.

RESULT = Q9SCR(PRINTW, FORMAT, VALUE, ...)

The PRINTW function provides for formatted printing to a window. The FORMAT and VALUE arguments are passed to Q9SPRINT for conversion to a character string which is then added to the current window as if by an ADDSTR call.

See the documentation for Q9SPRINT for a description of the FORMAT and VALUE arguments. Note: the first argument to Q9SPRINT is a buffer for the formatted result text. On a PRINTW call, Q9SCR allocates this buffer for the user from the dynamic stack.

LEN = Q9SCR(REFRESH, RETURN-BUFFER, TERMINAL-TYPE)

Unless a window was created with the OPDIR option and it is being monitored by way of the MCU DTA or DTN commands, none of the Q9SCR functions actually change the terminal screen until REFRESH is called. Instead, each window has a private text buffer where changes are recorded. Q9SCR also keeps a buffer for each screen which is what Q9SCR thinks the terminal screen looks like.

When REFRESH is called, Q9SCR compares the window buffer to the screen buffer to see what changes need to be made to the screen to make it match the window in the region of the screen occupied by the window. The commands and text necessary to effect this match are placed in the RETURN-BUFFER supplied by the caller, and the screen buffer is updated. It is the caller's responsibility to actually send the RETURN-BUFFER to the terminal (for example, with a Q5SNDMJC call).

REFRESH returns the count of characters placed in the RETURN-BUFFER. This count is normally passed on to Q5SNDMJC or Q5PUTP. Note that Q5SNDMJC and Q5SNDMCR silently truncate messages after 2000 characters, so the application may need to send the contents of RETURN-BUFFER in parts.

Q9SCR does no checking for overflowing the RETURN-BUFFER, so the application should make sure it is large enough. Generally, a buffer size of:

$$\text{SIZE} = \text{ROWS} * (\text{COLUMNS} + 10) + 20$$

should suffice, where SIZE is in characters, and ROWS and COLUMNS are the dimensions of the screen.

Since the escape sequences necessary to control a terminal vary from device to device, Q9SCR needs to know what kind of terminal it should generate control codes for. The application supplies the type by way of the TERMINAL-TYPE argument. The currently known terminal types are listed in table 11-1. More types can be added by your system administrators.

The terminal type of zero (0) requests that the entire contents of the screen (after updating) be copied to the RETURN-BUFFER. Each screen line will end with a VSOS end-of-line (#1F) and will have had trailing spaces removed. The RETURN-BUFFER is thus in the correct form to be written to an R-type file as a screen snapshot or sent to a terminal of unknown type.

If a window created with the OPDIR option is REFRESHed, the return buffer is filled as if the TERMINAL-TYPE were 0, regardless of the value supplied on the call. This is because the screen buffer and the window buffer are identical, so there is no way Q9SCR can compare the two to determine the changes. But note that if the window's only purpose is to be available to the MCU, then the MCU handles the details of refreshing the terminal, and REFRESH calls are neither necessary nor useful, except to get snapshots.

WPTR = Q9SCR(RESET, SCREEN-NUMBER)

The RESET function is used to destroy a screen and create it afresh. All memory associated with the screen is freed and then the window is created again, as if as the result of an INITSCR call.

RESET returns the pointer for the new standard window for the screen.

RESULT = Q9SCR(SCROLL)

The SCROLL function scrolls the entire window up one line. That is, the current top line is deleted, the remaining lines move up one line, and the new bottom line is filled with spaces.

The cursor position in the window does not change, which means it moves down a line relative to the window contents.

The SCROLL function is legal even if scrolling is normally disabled on the window.

RESULT = Q9SCR(SCROLLOK, FLAG)

A window can be set to scrolling or non-scrolling. When a character is written to the last column of the last line of a scrolling window, the contents of the window are moved up a line and a new, blank line is created as the bottom line. The previous top line is discarded. The cursor normally moves to the first column of this new line.

If the last character of the last line of a non-scrolling window is written, Q9SCR does not scroll. Instead, the cursor stays where it was and an ESCROLL error is returned.

The SCROLLOK function sets whether a window is scrolling or non-scrolling. If FLAG is zero, scrolling is disabled. If FLAG is non-zero, then scrolling is permitted. New windows are non-scrolling by default.

WPTR = Q9SCR(SUBWIN, ROWS, COLS, S-ROW, S-COL)

Sometimes it is useful to restrict modifications to a portion of a window or to logically partition a window into separate pieces. The SUBWIN function returns a pointer to a subwindow of the current window. The current window is called the parent of the subwindow. The upper, left corner of the subwindow is at coordinates S-ROW, S-COL relative to its parent, and the subwindow's size in ROWS rows by COLS columns. If ROWS or COLS is zero (0) then SUBWIN will set it to the largest value compatible with the size of the parent and S-ROW and S-COL.

A subwindow must fit entirely within its parent. The subwindow and parent window share the same text buffer, so any change to the subwindow directly affects the parent window. Similarly, any change to the parent window in the region occupied by the subwindow also changes the subwindow.

Subwindows move with their parent windows and are destroyed when their parents are deleted. In addition, a subwindow may be moved or deleted directly with the MWIN and DELWIN functions.

LIMITS

The Q9SCR routines are not direct replacements for the Unix† routines with similar names, so care must be taken in porting a Unix curses program to the VSOS environment. In most cases, the functionality is the same, but the default values and argument orders may be different.

Table 11-2 shows the maxima and minima for Q9SCR:

Table 11-2. Q9SCR Maxima and Minima

Limit	Min	Max
Columns per line	8	136
Rows per window	2	100
Screens	1	10
Size of PRINTW result string	0	one window full

†Unix is a trademark of AT&T Bell Laboratories.

Q9SPRINT

Q9SPRINT is a routine which implements most of the Unix† sprintf(3S) functionality to make it easier for an application, especially a non-FTN200 application, to create formatted text strings.

USAGE

Q9SPRINT takes a format string and zero or more arguments whose values are converted to a displayable form based on the format. The resultant string is returned in a buffer supplied by the caller.

Q9SPRINT is called as:

```
LEN = Q9SPRINT(RS, FORMAT, VALUE1, VALUE2, ... VALUEN)
```

where:

LEN Is the length (in bytes) of the result string.

RS Is a buffer to receive the formatted result string. It is up to the user to make sure RS is big enough. For FTN200 character variables only, Q9SPRINT can determine the result buffer length and will truncate the result rather than overflow the variable.

FORMAT Is a character string containing formatting directives.

VALUE1, ... VALUEN Are the arguments to be converted according to the format.

The format contains both text which is copied directly to the result string and formatting directives. A formatting directive starts with an escape character, normally %, and has the form:

```
<e><j><Z><W><.p><C>
```

where each of the bracketed letters stands for a specific character, character pair, or numeric string, appearing in the order listed, and with the following meanings. The <e> and <c> tokens are required. The rest are optional.

<e> An escape character to indicate the start of a formatting directive. The default character is a percent sign. It may be changed by:

```
CALL Q9ESCCH (NEWCH)
```

where NEWCH is the new escape character. For instance, to use an ampersand instead of a percent sign, execute the statement:

```
CALL Q9ESCCH (1R&)
```

<j> A minus sign (-) to left justify the result. Omitted to right justify.

<Z> The digit 0 to zero fill numeric conversions. Omitted to space fill.

†Unix is a trademark of AT&T Bell Laboratories.

- <w> A numeric value for the field width. The conversion will be right justified (left justified if <j> is specified) in a field <w> characters wide. If the conversion will not fit, the field size is increased until it will. If omitted, the field will be the minimum width necessary to contain the converted value.
- <.p> A period (.) followed by a numeric value for the precision. For E and F conversions, this is the number of digits to follow the decimal point (default is 6). For string conversions, it is the maximum number of characters to copy.
- <c> A one- or two-character format selector (either upper or lower case may be used) which specifies how the next argument (VALUE1) is to be converted to ASCII for inclusion in the result string. The selectors are listed in the following table. Many of the conversions produce results very similar to FTN200 formatted output conversions. When this is true, the FTN200 edit descriptor is also shown.

<c>	FTN200	Argument Value
D	I<w>	Decimal integer (only the low 48 bits are used).
O		Octal integer.
X	Z<w>	Hexadecimal integer.
F	F<w><.p>	Full-word floating point number.
E	E<w><.p>	Full-word floating point number to be converted to scientific notation.
G	G<w><.p>	Full-word floating point to be converted using E or F format, whichever gives full precision in the minimum space.
C		Character, right justified in a word (for example, 1RX).
S	A<w>	Zero byte terminated character string. At most 65535 characters will be processed. If the corresponding argument is an FTN200 character variable or literal, the zero byte is not needed and the string has its natural length.
/		Delimited character string. The first character of the string is taken to be a delimiter. Up to 65535 characters following the first character until the delimiter is seen again will be put in the result string.
=x		Delimited character string. Like /, except the delimiter character (x) is given by the next character in FORMAT, rather than the first character of the string. Most often, x is a space, for example %=, in order to copy the string up to the first blank.

<c>	FTN200	Argument Value
N	/	Replaced by a newline (#lf) in the result string.
\$		Marks the end of the format string. A zero byte also terminates the format string. IMPL callers will almost always need to end their format strings with %\$.
x		Any other character is output as is, without the leading %, so a single % is indicated by %%.

<w> or <p> may be an asterisk (*) instead of a numeric value. In this case, the next argument is taken to be the value. If the value is negative, the default for <w> or <p> is used.

EXAMPLES

```
INTEGER ANSWER, F
REAL FANSWER
DIMENSION RS (10)
DATA ANSWER / 1234 /
DATA FANSWER / 1234.567 /
```

```
* F = Q9SPRINT (RS, 'THE ANSWER IS %D', ANSWER)
  RS = 'THE ANSWER IS 1234'

* F = Q9SPRINT (RS, 'THE ANSWER IS %-10D', ANSWER)
  RS = 'THE ANSWER IS 1234'

* F = Q9SPRINT (RS, 'THE ANSWER IS %-10D', ANSWER)
  RS = 'THE ANSWER IS 1000000234'

* F = Q9SPRINT (RS, 'THE ANSWER IS %0*D', ANSWER)
  RS = 'THE ANSWER IS 1000000234'

* F = Q9SPRINT (RS, 'THE ANSWER IS %F', FANSWER)
  RS = 'THE ANSWER IS 1234.567000'

* F = Q9SPRINT (RS, 'THE ANSWER IS %10.F', FANSWER)
  RS = 'THE ANSWER IS 1234.567000'

* F = Q9SPRINT (RS, 'THE ANSWER IS %/' , '#A TEXT STRING#')
  RS = 'THE ANSWER IS A TEXT STRING'
```

For IMPL, the last example would become: (note the %\$)

```
* F = Q9SPRINT (RS, 'THE ANSWER IS %/'%$, #A TEXT STRING#')
  RS = 'THE ANSWER IS A TEXT STRING'
```

ALTERNATE INTERFACES

Two additional routines make using Q9SPRINT from an IMPL or FTN200 application easier: Q9PUTLN, which calls Q5PUTN, and Q9FPUTLN, which writes to an FTN200 logical unit.

Q9PUTLN is called as:

```
STATUS = Q9PUTLN(FLUN, FORMAT, VALUE1, ... VALUEN)
```

Q9FPUTLN is called as:

```
CALL Q9FPUTLN(UNIT, FORMAT, VALUE1, ... VALUEN)
```

The FORMAT and VALUE arguments are the same as the Q9SPRINT. For Q9PUTLN, the FLUN argument is the file logical unit number of an open SIL file to which the formatted string is written. Alternatively, if FLUN is a small, negative integer, the string is output using one of the Q5SNDMxx calls as follows:

FLUN Value	Call	Description
- 1	Q5SNDMCR	Send to task controller
- 2	Q5SNDMCE	Send to task controllee
- 3	Q5SNDMJC	Send to job controller
- 4	Q5SNDMDF	Send to job dayfile
- 5	Q5SNDMOP	Send to operator

Q9PUTLN returns the SIL error code from the Q5PUTP or Q5SNDMxx call.

On the Q9FPUTLN call, the UNIT argument is an FTN200 logical unit number (for example, to write to TAPE6) or a file name (for example, 'OUTPUT'). Thus, Q9FPUTLN output can be mixed with normal FTN200 output in the same file.

MISCELLANEOUS DEFINITIONS AND GUIDELINES

This section contains the following:

- A definition of the format of the display table that is required for screen displays that may be viewed by the DTA and DTN commands (see the VSOS Operator's Guide for descriptions of these commands).
- Guidelines to be followed in implementing display formatting programs.

FORMAT OF DISPLAY TABLE

The format of display manager controlled display tables is as follows:

Word 1 Flag signifying that this is a display table. This must contain the following:

8HACTIVE	If the display program is currently executing or
8HINACTIVE	If the display program is not currently executing.

Word 2 Reserved for future use except for a field in bits 56 - 63. This must be set to one to indicate that the data is formatted with no embedded control characters for use as display manager displays.

Word 3 - 202 Contains the formatted display - 20 lines at 80 characters per line.

Display tables residing in virtual memory must be wholly confined within a 512 word block of memory through it does not have to be aligned with the beginning of a block.

GUIDELINES FOR IMPLEMENTING TABLE DISPLAYS

The display tables that reside in VSOS shared table space shall be referred to as table displays. For the programs that perform the formatting of these displays the following guidelines should be followed:

- Upon initial execution of the formatting code, the following should be done:
 - Set the values for first two words as indicated above.
 - Format the display with initial data.
- Upon each subsequent execution of the formatting code, do the following:
 - Find the display's entry in T_DSPTSK (may be defined by a compiled parameter value if various displays are permanently assigned to specific T_DSPTSK entries). If either of the following conditions obtain, then refresh/reformat the display data and clear the remop bit of the flags field of the T_DSPTSK entry:
 - One of the three MCU display active fields in MISCTAB is set to the index of the T_DSPTSK entry.
 - The remop bit of the flags field of the T_DSPTSK entry is set.

GUIDELINES FOR IMPLEMENTING DISPLAY TASKS

Controllees that generate displays whose data resides in their virtual space will be referred to as display tasks. In implementing these displays, the following guidelines should be followed:

- Have the first two words of the data table preset to 8HINACTIVE and 1, respectively.
- Send a system message to the display manager to create a display table entry (message option = 2).
- Fill in the initial set of display data.
- Change the first word of the display table to 8HACTIVE.
- Repeatedly update the display data. (This step might include some method of giving up processing time to minimize time spent in the controllee.)
- If the display program has a way to determine that it can quit and disappear, then it should send a system message to the display manager to delete its table entry (message option = 3).

CHARACTER SET

A

The ASCII character set is shown in table A-1. Aids for hexadecimal-to-octal and hexadecimal-to-decimal conversion are given in tables A-2 and A-3.

Table A-1. American National Standard Code for Information Interchange (ASCII)
With Punched Card Codes and EBCDIC Translation

		0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
		0	1	2	3	4	5	6	7	8	9	10 (A)	11 (B)	12 (C)	13 (D)	14 (E)	15 (F)
b4 b3 b2 b1	COL ROW																
0 0 0 0	0	NUL 12-0-9-8-1 NUL 00	DLE 12-11-9-8-1 DLE 10	SP no-punch SP 40	0 0 F0	@ 8-4 @7C	P 11-7 P D7	\ 8-1 79	p 12-11-7 p 97	11-0-9-8-1 DS 20	12-11-0-9-8-1 30	12-0-9-1 41	12-11-9-8 58	12-11-0-9-6 76	12-11-8-7 9F	12-11-0-8 8B	12-11-9-8-4 DC
0 0 0 1	1	SOH 12-9-1 SOH 01	DC1 11-9-1 DC1 11	1 12-8-7 4F	1 1 F1	A 12-1 A C1	Q 11-8 Q D8	a 12-0-1 a 81	q 12-11-8 q 98	0-9-1 SOS 21	9-1 31	12-0-9-2 42	11-8-1 59	12-11-0-9-7 77	11-0-8-1 A0	12-11-0-9 B9	12-11-9-8-5 DD
0 0 1 0	2	STX 12-9-2 STX 02	DC2 11-9-2 DC2 12	8-7 7F	2 2 F2	B 12-2 B C2	R 11-9 R D9	b 12-0-2 b 82	r 12-11-9 r 99	0-9-2 FS 22	11-9-8-2 1A	12-0-9-3 43	11-0-9-2 62	12-11-0-9-8 78	11-0-8-2 AA	12-11-0-8-2 BA	12-11-9-8-6 DE
0 0 1 1	3	ETX 12-9-3 ETX 03	DC3 11-9-3 DC3 13	# 8-3 # 7B	3 3 F3	C 12-3 C C3	S 0-2 S E2	c 12-0-3 c 83	s 11-0-2 s A2	0-9-3 23	9-3 33	12-0-9-4 44	11-0-9-3 63	12-0-8-1 80	11-0-8-3 AB	12-11-0-8-3 BB	12-11-9-8-7 DF
0 1 0 0	4	EOT 9-7 EOT 37	DC4 9-8-4 DC4 3C	\$ 11-8-3 \$ 5B	4 4 F4	D 12-4 D C4	T 0-3 T E3	d 12-0-4 d 84	t 11-0-3 t A3	0-9-4 BYP 24	9-4 PN 34	12-0-9-5 45	11-0-9-4 64	12-0-8-2 8A	11-0-8-4 AC	12-11-0-8-4 BC	11-0-9-8-2 EA
0 1 0 1	5	ENQ 0-9-8-5 ENQ 2D	NAK 9-8-5 NAK 3D	% 0-8-4 % 6C	5 5 F5	E 12-5 E C5	U 0-4 U E4	e 12-0-5 e 85	u 11-0-4 u A4	11-9-5 NL 15	9-5 RS 35	12-0-9-6 46	11-0-9-5 65	12-0-8-3 8B	11-0-8-5 AD	12-11-0-8-5 BD	11-0-9-8-3 EB
0 1 1 0	6	ACK 0-9-8-6 ACK 2E	SYN 9-2 SYN 32	& 12 & 50	6 6 F6	F 12-6 F C6	V 0-5 V E5	f 12-0-6 f 86	v 11-0-5 v A5	12-9-6 LC 06	9-6 UC 36	12-0-9-7 47	11-0-9-6 66	12-0-8-4 8C	11-0-8-6 AE	12-11-0-8-6 BE	11-0-9-8-4 EC
0 1 1 1	7	BEL 0-9-8-7 BEL 2F	ETB 0-9-6 ETB 26	8-5 7D	7 7 F7	G 12-7 G C7	W 0-6 W E6	g 12-0-7 g 87	w 11-0-6 w A6	11-9-7 IL 17	12-9-8 GE 08	12-0-9-8 48	11-0-9-7 67	12-0-8-5 8D	11-0-8-7 AF	12-11-0-8-7 BF	11-0-9-8-5 ED
1 0 0 0	8	BS 11-9-6 BS 16	CAN 11-9-8 CAN 18	(12-8-5 (4D	8 8 F8	H 12-8 H C8	X 0-7 X E7	h 12-0-8 h 88	x 11-0-7 x A7	0-9-8 28	9-8 38	12-8-1 49	11-0-9-8 68	12-0-8-6 8E	12-11-0-8-1 B0	12-0-9-8-2 CA	11-0-9-8-6 EE
1 0 0 1	9	HT 12-9-5 HT 05	EM 11-9-8-1 EM 19) 11-8-5) 5D	9 9 F9	I 12-9 I C9	Y 0-8 Y E8	i 12-0-9 i 89	y 11-0-8 y A8	0-9-8-1 29	9-8-1 39	12-11-9-1 51	0-8-1 69	12-0-8-7 8F	12-11-0-1 B1	12-0-9-8-3 CB	11-0-9-8-7 EF
1 0 1 0	10 (A)	LF 0-9-5 LF 25	SUB 9-8-7 SUB 3F	* 11-8-4 * 5C	8-2 7A	J 11-1 J D1	Z 0-9 Z E9	j 12-11-1 j 91	z 11-0-9 z A9	0-9-8-2 SM 2A	9-8-2 3A	12-11-9-2 52	12-11-0 70	12-11-8-1 90	12-11-0-2 B2	12-0-9-8-4 CC	12-11-0-9-8-2 I(LVM) FA
1 0 1 1	11 (B)	VT 12-9-8-3 VT 0B	ESC 0-9-7 ESC 27	+ 12-8-6 + 4E	11-8-6 5E	K 11-2 K D2	12-8-2 4A	k 12-11-2 k 92	12-0 C0	0-9-8-3 CU2 2B	9-8-3 CU3 3B	12-11-9-3 53	12-11-0-9-1 71	12-11-8-2 9A	12-11-0-3 B3	12-0-9-8-5 CD	12-11-0-9-8-3 FB
1 1 0 0	12 (C)	FF 12-9-8-4 FF 0C	FS 11-9-8-4 IFS 1C	< 0-8-3 < 6B	12-8-4 4C	L 11-3 L D3	0-8-2 E0	l 12-11-3 l 93	12-11 6A	0-9-8-4 2C	12-9-4 PF 04	12-11-9-4 54	12-11-0-9-2 72	12-11-8-3 9B	12-11-0-4 B4	12-0-9-8-6 CE	12-11-0-9-8-4 FC
1 1 0 1	13 (D)	CR 12-9-8-5 CR 0D	GS 11-9-8-5 IGS 1D	- 11 - 60	8-6 7E	M 11-4 MD4	11-8-2 5A	m 12-11-4 m 94	11-0 D0	12-9-8-1 RLF 09	11-9-4 RES 14	12-11-9-5 55	12-11-0-9-3 73	12-11-8-4 9C	12-11-0-5 B5	12-0-9-8-7 CF	12-11-0-9-8-5 FD
1 1 1 0	14 (E)	SO 12-9-8-6 SO 0E	RS 11-9-8-6 IRS 1E	> 12-8-3 > 4B	0-8-6 6E	N 11-5 ND5	11-8-7 5F	n 12-11-5 n 95	11-0-1 A1	12-9-8-2 SMM 0A	9-8-6 3E	12-11-9-6 56	12-11-0-9-4 74	12-11-8-5 9D	12-11-0-6 B6	12-11-9-8-2 DA	12-11-0-9-8-6 FE
1 1 1 1	15 (F)	SI 12-9-8-7 SI 0F	US 11-9-8-7 IUS 1F	/ 0-1 / 61	0-8-7 6F	O 11-6 OD6	0-8-5 6D	o 12-11-6 o 96	DEL 12-9-7 DEL 07	11-9-8-3 CU1 1B	11-0-9-1 E1	12-11-9-7 57	12-11-0-9-5 75	12-11-8-6 9E	12-11-0-7 B7	12-11-9-8-3 DB	EO 12-11-0-9-8-7 FF

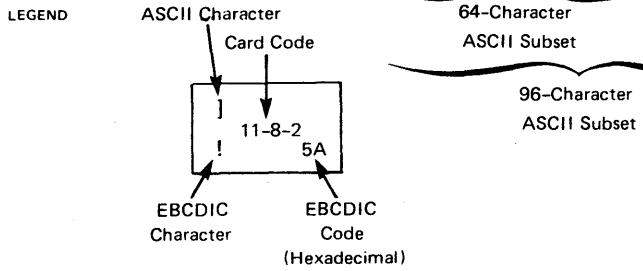


Table A-2. Hexadecimal-Octal Conversion

		First Hexadecimal Digit (Leftmost of a 2-digit number)															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Second Hexadecimal Digit (Rightmost of a 2-digit number)	0	000	020	040	060	100	120	140	160	200	220	240	260	300	320	340	360
	1	001	021	041	061	101	121	141	161	201	221	241	261	301	321	341	361
	2	002	022	042	062	102	122	142	162	202	222	242	262	302	322	342	362
	3	003	023	043	063	103	123	143	163	203	223	243	263	303	323	343	363
	4	004	024	044	064	104	124	144	164	204	224	244	264	304	324	344	364
	5	005	025	045	065	105	125	145	165	205	225	245	265	305	325	345	365
	6	006	026	046	066	106	126	146	166	206	226	246	266	306	326	346	366
	7	007	027	047	067	107	127	147	167	207	227	247	267	307	327	347	367
	8	010	030	050	070	110	130	150	170	210	230	250	270	310	330	350	370
	9	011	031	051	071	111	131	151	171	211	231	251	271	311	331	351	371
	A	012	032	052	072	112	132	152	172	212	232	252	272	312	332	352	372
	B	013	033	053	073	113	133	153	173	213	233	253	273	313	333	353	373
	C	014	034	054	074	114	134	154	174	214	234	254	274	314	334	354	374
	D	015	035	055	075	115	135	155	175	215	235	255	275	315	335	355	375
	E	016	036	056	076	116	136	156	176	216	236	256	276	316	336	356	376
	F	017	037	057	077	117	137	157	177	217	237	257	277	317	337	357	377
Octal		000 - 037	040 - 077	100 - 137	140 - 177	200 - 237	240 - 277	300 - 337	340 - 377								

Table A-3. Hexadecimal-Decimal Conversion

		Exponent for Base 16					
		5	4	3	2	1	0
Hexadecimal Number	0	0	0	0	0	0	0
	1	1048576	65536	4096	256	16	1
	2	2097152	131072	8192	512	32	2
	3	3145728	196608	12288	768	48	3
	4	4194304	262144	16384	1024	64	4
	5	5242880	327680	20480	1280	80	5
	6	6291456	393216	24576	1536	96	6
	7	7340032	458752	28672	1792	112	7
	8	8388608	524288	32768	2048	128	8
	9	9437184	589824	36864	2304	144	9
	A	10485760	655360	40960	2560	160	10
	B	11534336	720896	45056	2816	176	11
	C	12582912	786432	49152	3072	192	12
	D	13631488	851968	53248	3328	208	13
	E	14680064	917504	57344	3584	224	14
	F	15728640	983040	61440	3840	240	15

$$\begin{array}{c|c} & i \\ \hline j & m \end{array}$$

$j_{16} \times 16^i = m_{10}$

To find $E_{16} \times 16^3$; look at row E, column 3 and find 57344

MESSAGES

B

This appendix lists messages that system routines send to the job dayfile or interactive terminal. Each dayfile message is preceded by the time the message was written.

Error messages from privileged system tasks occasionally appear in a job dayfile. To determine the meaning of a privileged system task error code, refer to appendix B in the VSOS 2 Operator's Guide.

Table B-1 lists the messages from the system utilities described in this manual. Each message is listed with its return code value. When an error occurs, the system compares the return code value with the threshold value specified on the TV control statement to determine whether the job aborts because of the error.

Code interpretation is as follows:

- 0 No error
- 4 Warning (nonfatal) error
- 8 Fatal error

Table B-2 lists SIL, BATCHPRO and RHF messages. Each message consists of a maximum of 136 characters and has the following format:

severity	routine	code	description
severity			Error severity, either fatal (F), warning (W), or informative (I). If the STATUS=,stat parameter on the SIL call is not specified, SIL returns the appropriate return code (8, 4, or 0) to the task's controller.
routine			Name of the SIL routine that detected the error. Table B-2 lists the routines that can issue the error.
code			Status code. Table B-2 lists the messages in order of their status codes.
description			Description of the error.

When a fatal SIL error message is sent to the job dayfile or interactive terminal, it is also sent to the system dayfile. A message sent to the system dayfile is prefixed by the user number and controllee file name.

Table B-3 lists the meanings of system error codes returned if the system quits during program execution.

Table B-4 lists the meanings of the tape error codes returned in the SIL error message for status code 1476. The tape error code is also stored in the ioer field of the FIT. After receiving a status code of 1476, the caller can issue a Q5GETFIT call with the IOER= parameter specified to get the tape error code from the FIT.

System utilities accept the returned SIL status and, if necessary, modify the severity (first character) of the error message to be consistent with their operations. Thus, if the utility produces a return code of 8, the corresponding error should appear in the dayfile as a fatal message and should be modified accordingly. If the return code is 0 or 4, the error should appear as a warning. If the return code is 0, it should appear as informative.

Table B-1. Diagnostic Messages (Sheet 1 of 24)

Message	Significance/Action	Return Code	Issued By
ACCESS CONFLICT WITH ANOTHER JOB ON FILE	The file cannot be opened because another job has the file open in an filename access mode that conflicts with the requested access mode. Wait until the file is free and try again.	8	VSOS
ACCESS VIOLATION ON FILE filename	The request failed because the user does not have the required access permission to the indicated file. If possible, add the required access permission to the access permission set for the file and try again.	-	VSOS
ADDR=REGnn	Address indicated by the symbol specified corresponds to register nn.	4	DEBUG
ADR IN REG FILE	A location, or the computation of name=location, was less than #4000. The user cannot reference the virtual address range 0 through #3FFF.	4	DEBUG
ADR NOT COD ADR	Address specified in EXECUTE, BKPT, or BKPTR command is not that of executable code. No action will be taken.	4	DEBUG
ADR BEYOND MOD	Address specified does not fall within the current module.	4	DEBUG
ALL DONE	The system has completed the task.	-	VSOS
ALPHA OUT OF BOUNDS	Address of Alpha message is out-of-bounds.	-	VSOS
ATTACHED NEWPL TOO SMALL	The NEWPL attached to the job is too small and cannot be extended. Rerun with larger file.	8	UPDATE
ATTEMPT TO READ PAST EOF ON FILE	Do not attempt to read past end of file.	-	VSOS
ATTEMPTED READ WITH INVALID FILE NUMBER - nnnn	Notify a systems analyst.	8	UPDATE
ATTEMPTED TO READ FROM UNKNOWN FILE	Correct READ statement.	8	UPDATE
BAD BINARY FILE filename THE WORD 'MODULE' NOT FOUND	Self-explanatory.	8	LOAD

Table B-1. Diagnostic Messages (Sheet 2 of 24)

Message	Significance/Action	Return Code	Issued By
BAD CALL DIRECTIVE	Invalid deck name on CALL directive. Correct and rerun.	8	UPDATE
BAD CARD ENCOUNTERED	Correct input file card.	8	UPDATE
BAD COMMAND	Correct DEBUG directive.	4	DEBUG
BAD DFM IN CTEE	The controllee to be debugged had been loaded with only a portion of the Data Flag Manager (DFM). When the controllee contains DFM, DEBUG links to the controllee's copy of DFM. When the controllee does not contain DFM, DEBUG uses its own copy. However, when the controllee contains only part of the DFM, DEBUG assumes something is wrong and aborts. The controllee should contain all or none of the following module/entry points: FT_SYSTEM/Q7DFINIT; DFBH_/Q7DFCL1.	8	DEBUG
BAD FILE NAME FOR READ - filename	Invalid filename on a *READ directive. Correct and resubmit.	8	UPDATE
BAD FUNCTION CODE FOR GET/SEND MESSAGE	Notify a systems analyst.	8	UPDATE
BAD LIB FILE filename THE WORD 'LIBRARY' NOT FOUND	Self-explanatory.	8	LOAD
BAD LOGIN	A parameter on the LOGIN line is incorrect.	-	VSOS
BAD MINUS PAGE	Minus page of the task is not set up correctly.	-	VSOS
BAD NAME, NO NAME OR DUPLICATE IDENT	Correct UPDATE directive.	8	UPDATE
BAD ORIGIN ADDRESS FOR GROUP	The specified virtual bit address must be on a small page or large page boundary.	8	LOAD
BATCHPRO PROBLEM -- CHECK FOR DROP FILE	BATCHPRO not found or aborted. Check a BATCHPRO drop file under user's user number, run DUMP on the drop file to help find the problem; notify a systems analyst.	-	VSOS
BKPT NOT FOUND	Correct BKPTR directive.	4	DEBUG

Table B-1. Diagnostic Messages (Sheet 3 of 24)

Message	Significance/Action	Return Code	Issued By
BKPT TABLE FULL	No additional breakpoints can be set until one or more existing breakpoints are removed.	4	DEBUG
BOUND IMPLICIT MAP ANOMALY	Garbage in the minus page. Rebuild the minus page.	-	VSOS
CALL OF FOLLOWING COMDECK IS RECURSIVE - name	Convert call or calls and resubmit.	8	UPDATE
CALL OF INACTIVE OR PURGED COMDECK	Informative message.	8	UPDATE
CALL OF NON COMDECK	Change call to existing common deck name.	8	UPDATE
CANNOT GROS/GROL A PRESET COMMON BLOCK	Load has detected loader text that is presetting a common block that has no controllee file space.	8	LOAD
CANNOT RESTART - REGTBL full	Cannot restart a drop file. System table REGTBL is full. Try again later.	-	VSOS
CANT CREATE SCRATCH FILE TO USE COMDECK OUT OF ORDER	Not enough mass storage available for handling out of order comdecks. Rerun later.	8	UPDATE
CANT DESTROY EXISTING DROP FILE	Modified pages have been written to the task drop file; therefore, the file cannot be purged.	-	VSOS
CENTRAL MEMORY PARITY ERROR	Rerun job.	-	VSOS
CHARACTER STRING FOUND AT nnnnnnnnnnn	The SEARCH directive was processed successfully; LOOK found the character string at the specified address.	-	LOOK
CHARACTER STRING FOUND ONLY n TIMES	LOOK encountered the end of file before it found the specified number of occurrences of the string. It found n occurrences.	-	LOOK
CHARACTER STRING NOT FOUND	LOOK encountered the end of file before it found an occurrence of the string.	-	LOOK
CHARGE STATEMENT MUST BE SUPPLIED	User must execute a CHARGE statement.	-	VSOS
COMDECK ARRAY TOO SMALL	Too many comdecks for the comdeck array.	8	UPDATE
COMDECK DOES NOT EXIST - name	Convert call and resubmit.	8	UPDATE

Table B-1. Diagnostic Messages (Sheet 4 of 24)

Message	Significance/Action	Return Code	Issued By
reason - COMMAND IGNORED	When the parameters for a LOOK directive are meaningless, missing, or illegal, the directive is ignored. Refer to the reason given for more information.	4	LOOK
COMMON BLOCK TABLE OVERFLOW	Loader table overflow. Contact a site analyst and request that the loader be rebuilt with a larger COMXR table.	4	LOAD
COMPARE TERMINATED - END OF FILE filename	Informative message.	-	COMPARE
CONSECUTIVE DELIMITERS IN EXECUTE LINE	The user cannot specify consecutive parameter delimiters(two or more left parentheses or commas) on a LOAD control statement. Reenter the corrected statement.	8	LOAD
CONTRADICTION, LIST=0 AND OUTPUT=	Correct control statement.	8	OLE
CONTROLLEE FILE filename IS TOO SMALL, NEED nn(#mm) BLOCKS	Rerun with a larger controllee file specified. nn indicates the number of blocks needed; mm indicates the hexadecimal number.	8	LOAD
CONTROLLEE REQUIRES SHARED LIB	Controllee needs an active shared library to execute.	-	LOAD
CONTROLLEE FORMAT ERROR	Error in format of the controllee option.	8	LOAD
CONTROLLEE MUST BE RELOADED	The controllee is loaded in the format used by a previous system release and is not compatible with this system. Reload the controllee and try again.	-	VSOS
CONTROLLEE USING WRONG LIBRARIES	The controller is running with libraries other than those with which it was built.	4	Q5INIT Q5INITCH
COULD NOT CREATE NEWPL	No mass storage space available. Rerun job.	8	UPDATE
COULD NOT LINK CORRECTION TO DIRECTORY ENTRY	Notify a systems analyst.	8	UPDATE
CREATION RUN ABORT	One or more fatal errors were encountered in a run. Correct errors and rerun creation.	8	UPDATE
DECK DOES NOT EXIST - name	Correct UPDATE directive.	4	UPDATE

Table B-1. Diagnostic Messages (Sheet 5 of 24)

Message	Significance/Action	Return Code	Issued By
DECK/DIREC LENGTH ERROR IN UPDREST. NO NEWPL CREATED	Notify a systems analyst.	8	UPDATE
DECKS OR IDENTs OUT OF ORDER	Ending deck specified in a range precedes the beginning deck on the oldpl.	4	UPDATE
DEVnnn BLOCK COUNT=nnnn	Informative message providing the absolute block count. The absolute block count is the number of tape blocks read or written from the beginning of the tape volume.	-	-
DEVnnn BLOCK COUNT MISMATCH CURRENT BLOCK COUNT=nnnn LABEL BLOCK COUNT=nnnn	The block count in the EOF1 or EOVI label does not match the current block count kept for the file or volume.	-	-
DOUBLE TRACK CORRECTABLE ERRORS=nnnn	Informative message providing the number of double-track correctable errors detected since the beginning of the tape volume. Additional detailed error information is recorded in the system error file accessible to site personnel.	-	-
DROP FILE HAS WRONG SMALL PG SIZE	The user attempted to restart a task by executing its drop file while VSOS is using a different small page size than it used when the drop file was created. Restart the task after a VSOS autoloader that selects the same small page size as that used by the drop file.	-	-
DROP FILE LENGTH IS TOO LONG	The drop file length in the minus page or FILEI is too large. Reload the controller or change the drop file length with TASKATT.	-	VSOS
DROP FILE MAP FULL, PAGE NOT MAPPED	The task has more than 170 noncontiguous common blocks. The user should attempt to group these blocks together (make them contiguous) using the page grouping LOADER directive. The task must be reloaded.	-	VSOS
DROP FILE OVERFLOW	Drop file cannot be extended. Use LOADER or TASKATT (DFL parameter in either case) to define a larger drop file size.	-	VSOS
DUMPING filename	Informative message identifying the file being dumped.	-	PFPRFIL
DUMPING filename USER=user POOL=pool	Informative message identifying the file being dumped and its owner.	-	PFPRFIL

Table B-1. Diagnostic Messages (Sheet 6 of 24)

Message	Significance/Action	Return Code	Issued By
EDITED CARDS WILL NOT BE RESTORED BY REMOVING - YYYY	Informative message. Some cards deactivated by ident YYYY have been permanently removed by editing.	4	UPDATE
EMPTY INPUT FILE IN Q MODE	In Q mode, the input file must contain at least a COMPILE directive. Correct and rerun.	8	UPDATE
ENCOUNTERED READ ERRORS OF OLDPL (BAD DATA)	The oldpl has been damaged or truncated and must be rebuilt from the previous version.	8	UPDATE
ENCOUNTERED UNPROCESSED MODIFICATIONS	Correct UPDATE directive.	8	UPDATE
ENDPROC filename	Processing of the specified procedure file has completed successfully.	-	BATCHPRO
ENTRY POINT TABLE OVERFLOW	Loader table overflow. Contact a site analyst and request that the loader be rebuilt with a larger ENTRYXR table.	4	LOAD
ERROR IN Q7PROMPT	Notify a systems analyst.	8	OLE
ERROR OPENING FILE filename	Q5OPEN failed to open file filename.	8	LOAD
ERROR OPENING LIBRARY FILE filename	Q5OPEN failed to open file filename.	8	LOAD
EXCEEDED DROP FILE MAP ENTRY LIMIT	Controllee requires more than 170 drop file map entries, use GRPALL=.	8	LOAD
EXCEEDED MAXIMUM SEQUENCE NUMBER	Maximum sequence number is 65,535.	8	UPDATE
EXECUTE FILE HAS WRONG SMALL PAGE SIZE	The user attempted to execute a task that was loaded with a page size smaller than the one currently running. Reload the task and try again.	-	VSOS
EXISTING PERMANENT FILE filename OPENED	File filename exists as an attached permanent file (informative message).	-	Q5GETFIL
EXISTING LOCAL FILE filename MADE PERMANENT	Informative message.	-	DEFINE
EXPECTED/BAD NAME ENCOUNTERED	Notify a systems analyst.	8	UPDATE
EXPECTED FILE NAME NOT ENCOUNTERED	Correct UPDATE directive.	8	UPDATE

Table B-1. Diagnostic Messages (Sheet 7 of 24)

Message	Significance/Action	Return Code	Issued By
EXPECTED IDENT NAME NOT FOUND	Correct UPDATE directive.	8	UPDATE
EXPECTED SEQUENCE NUMBER NOT FOUND	Correct UPDATE directive.	4	UPDATE
EXTENSION TRIED BEYOND VALIDATION, FILE filename	A routine attempted to write beyond the maximum file length validation for this user. Have the validation increased.	8	VSOS
EXTERNAL TABLE OVERFLOW	Loader table overflow. Contact a site analyst and request that the loader be rebuilt with a larger EXTXR table.	4	LOAD
FATAL SYSTEM ERROR	Rerun job.	-	VSOS
FILE filename param	SWITCH has changed a file characteristic for file filename as requested by parameter param.	-	SWITCH
FILE filename DOES NOT EXIST	No file assigned to the job has the name specified on the ROUTE statement.	8	ROUTE
FILE filename GIVEN TO POOL pool	The system successfully gave the specified file to the specified pool.	-	GIVE
FILE filename HAS BEEN RENAMED FILE newname	SWITCH has changed the file name from filename to newname.	-	SWITCH
FILE filename IS NOT A DYNAMIC/SHARED LIBRARY	The library specified by ULIB/SLIB is not a dynamic user library or a system shared library.	8	LOAD
FILE INDEX FULL. NONE OF YOUR PRIVATE FILES AVAILABLE.	Other users must destroy some of their files or log out to free space in the file index. Relogon (reenter the LOGON command after having previously issued a BYE) to bring in private files.	-	VSOS
FILE IS INCOMPLETE AND CANNOT BE EXECUTED	The executable file spans a device that is down.	-	VSOS
FILE NAMED filename NOT AN INPUT FILE	Correct control statement.	8	OLE
FILE filename NOT FOUND	One or more of the specified files does not exist. Correct the file name and try again.	8	LISTAC
FILE SEGMENT TABLE IS FULL	Rerun job.	-	VSOS

Table B-1. Diagnostic Messages (Sheet 8 of 24)

Message	Significance/Action	Return Code	Issued By
FILES COMPARED EQUALLY	Informative message.	-	COMPARE
FIRST FILENAME ILLEGAL - NO FILES PURGED	Self-explanatory.	8	PURGE
FOLLOWING CALLED COMDECK NOT FOUND - name	Correct call and resubmit.	8	UPDATE
FOLLOWING CARDS ARE SKIP- PED - NOT IN INSERT MODE	Informative message.	8	UPDATE
FORMAT ERROR	Correct control statement.	8	DEBUG EDITPUB
GENERATION OF UNIQUE IDNAME FAILED	Notify a systems analyst.	8	UPDATE
GROUP ORIGIN AT ADDRESS WHICH IS ALREADY ALLOCATED	An attempt was made to map a group of modules or common blocks to an address where another group of modules or common blocks is already mapped.	8	LOAD
HEX VALUE FOUND AT nnnnnnnnnnnn	The HSEARCH directive was processed successfully; LOOK found the hexadecimal value at the specified address.	-	LOOK
HEX VALUE FOUND ONLY n TIMES	LOOK encountered the end of file before it found the specified number of occur- rences of the value. It found n occurrences.	-	LOOK
HEX VALUE NOT FOUND	LOOK encountered the end of file before it found an occurrence of the value.	-	LOOK
IDENT DOES NOT EXIST	Correct UPDATE directive.	8	UPDATE
ILLEGAL BKPT	DEBUG has obtained control from an un- expected point in the controllee, namely, from a point at which DEBUG has not set a breakpoint.	4	DEBUG
ILLEGAL CHAR NUMBER	Correct hexadecimal number to include only digits 0 through 9 and letters A through F.	8	LOAD
ILLEGAL COMMAND	Illegal input parameters.	8	LOAD
ILLEGAL C504 REQUEST	User jobs cannot issue a C504 request.	-	VSOS

Table B-1. Diagnostic Messages (Sheet 9 of 24)

Message	Significance/Action	Return Code	Issued By
ILLEGAL DECK SPECIFIED IN MOVE. MOVE IGNORED	Correct UPDATE directive.	4	UPDATE
ILLEGAL FILE NAME FOR READ - filename	Invalid file name on a *READ directive. Correct and resubmit.	8	UPDATE
address ILLEGAL INSTRUCTION dropfile	Illegal instruction at bit address given.	-	VSOS
ILLEGAL LINK OPTION_option	You have specified an incorrect option on the LINK parameter. Option must be M, D, C, B, BC, or CB.	8	LOAD
ILLEGAL PARAMETER	Self-explanatory.	8	OLE
ILLEGAL RECORD TYPE FOR FILE outfil	The output file outfil has record type F or U. The user should rerun the task using either another output file or no output file.	8	DEBUG OLE
ILLEGAL REQUEST	Illegal Alpha function code.	-	VSOS
ILLEGAL USE OF FILE INPUT	The user cannot specify INPUT as the output file name in batch mode.	8	COPY COPYL
INADR EXCEEDS infile FILE LENGTH	I parameter is greater than the number of words in infile.	8	COPY
INTERACTIVE ACCESS NOT ALLOWED AT THIS TIME	The operator has turned off the INTRACTV job category, preventing interactive access to the system.	-	VSOS
INTERACTIVE ACCESS NOT PERMITTED	The user has not been granted interactive access to the system. Contact site personnel.	-	VSOS
INTERNAL ERROR. UNABLE TO RESTORE INACTIVE DECK	Notify a systems analyst.	8	UPDATE
INVALID CARD	Correct input.	8	UPDATE
INVALID DEVICE SET NAME devset	Specify correct device set name.	8	DMAP
INVALID DIRECTIVE FOR CREATION RUN	Only READ, DECK, COMDECK, and ADDFILE can appear in a creation run.	8	UPDATE
INVALID EXPRESSION: "expression"ENTER REPLACE- MENT OR CANCEL	Enter correct expression, enter CANCEL to abort, or carriage return to ignore bad expression.	8	Q7KEYWRD

Table B-1. Diagnostic Messages (Sheet 10 of 24)

Message	Significance/Action	Return Code	Issued By
INVALID FILE NAME	File names must be one through eight letters or digits beginning with a letter.	8	UPDATE
INVALID FILENAME	The specified file name is incorrect. A file name must be a string of one to eight letters or digits, beginning with a letter.	8	Q7KEYWRD
INVALID FILE OR FUNCTION	Correct UPDATE input.	8	UPDATE
INVALID FILE NUMBER OR FUNCTION	Correct UPDATE input.	8	UPDATE
INVALID PARAMETER xxx	The specified parameter is incorrect. Refer to the control statement description for more information.	8	Q7KEYWRD
address INVALID REF TO SHARED LIBRARY filename	The controllee has referenced a page in the library region that is not locked or mapped.	-	LOAD
INVALID RESOURCE CARD PARAMETER	A parameter specified on the RESOURCE statement is incorrect.	8	IQM
INVALID SEQUENCE NUMBER	Correct UPDATE directive.	8	UPDATE
INVALID TIME LIMIT	The user specified an incorrect time limit on the RESOURCE statement or on the execute line. The time limit must be a decimal integer between 0 and 599 940.	8	VSOS
INVALID UPDATE CHECK WORD	Notify a systems analyst.	8	UPDATE
INVALID USER NUMBER	User tried to use a reserved number.	-	VSOS
INVALID USER NUMBER	The specified user number is not valid. A CYBER 200 user number must be one to six decimal digits.	8	Q7KEYWRD
I/O ERROR RECEIVED BY WRPLY	Rerun job.	-	VSOS
IOC DOESN'T VERIFY	Invalid IOCs in a drop file being restarted. Usually means there is an IOC for a file that is no longer available.	-	VSOS
IOC FOR filename ALREADY IN USE	Self-explanatory.	8	DEBUG

Table B-1. Diagnostic Messages (Sheet 11 of 24)

Message	Significance/Action	Return Code	Issued By
JOB DROPPED BY OPERATOR	The operator entered a j.DROP command to drop this task.	-	BATCHPRO
JOB DROPPED BY USER	The user executed a DROP command for this task.	-	BATCHPRO
JOB FILE VACUOUS	The first record of a batch job must contain ASCII character control statements.	-	BATCHPRO
JOB KILLED BY OPERATOR	The operator entered a j.KILL command to kill this task.	-	BATCHPRO
JOB KILLED BY USER	The user executed a DROP command with the KILL option for this task.	-	BATCHPRO
LARGE PAGE LIMIT EXCEEDED DROFFILE	An instruction in the program requires more large pages than the current large page limit. Increase the large page limit. Address is the block address of the large page that the task faulted for.	-	VSOS
LARGE PAGE LIMIT EXCEEDS MAX WORKING SET	The specified large page limit exceeds the maximum working set for the job. Specify a smaller large page limit on the RESOURCE statement.	8	VSOS
LARGE PAGE LIMIT OF nnn PAGES EXCEEDED	The user specified a large page limit on the RESOURCE statement that exceeds the maximum memory available in the machine. Specify a large page limit not greater than nnn blocks.	8	VSOS
LENGTHS DON'T MATCH FOR COMMON BLOCK NAME ₁ IN MODULE NAME ₂	Name ₁ is the name of the common block, and name ₂ is the name of the module. The largest definition is allocated. A maximum of 10 of these error messages will be issued.	4	LOAD
LENGTHS DON'T MATCH FOR nn COMMON BLOCKS	If there are more than 10 conflicting common block definitions, this message is issued with total count of conflicting definition, nn.	4	LOAD
LIBRARY DIRECTORY AND INDEX TABLE FULL	Notify a systems analyst.	8	OLE
LIBRARY FILE filename IS EMPTY	The file filename has no modules on it.	8	LOAD

Table B-1. Diagnostic Messages (Sheet 12 of 24)

Message	Significance/Action	Return Code	Issued By
LOAD IS NOT USING THE SYSTEM SHARED LIBRARY	The shared library the loader is using to construct the controller is not the same one the system is using. This is because the user has a shared library file attached, has a pool attached containing a shared library file, or the system was autoloading using a shared library file not in the system pool or public set.	4	LOAD
LOADING filename	Informative message identifying file being loaded.	-	PFPRFIL
LOADING filename POOL=pool USER=user	Informative message identifying the file being loaded and its owner.	-	PFPRFIL
LOADMAP FORMAT ERROR	Bad input parameter for creating an output file.	8	LOAD
LOGIN FORMAT ERROR	The format of the LOGIN line is incorrect.	-	VSOS
MASTER CONTROL WORD DOES NOT MATCH OLDPL	Notify a systems analyst.	8	UPDATE
MAX WORKING SET LIMIT OF nnn BLOCKS EXCEEDED	The user specified a working set size limit on the RESOURCE statement that exceeds the maximum memory available in the machine. Specify a working set size limit not greater than nnn blocks.	8	VSOS
MAXIMUM ERRORS EXCEEDED	More than 256 errors. Correct and rerun.	8	UPDATE
MAXIMUM NUMBER OF CORRECTION HISTORY BYTES REACHED	Create new program library.	8	UPDATE
MAXIMUM NUMBER OF FIELDS EXCEEDED	Correct control statement.	8	UPDATE
MAXIMUM PERMISSIBLE NUMBER OF JOBS EXCEEDED	There is a limit to the number of jobs a user can have in the system. The user has attempted to put a job in the input queue that would exceed that limit. Wait for a job to complete.	8	VSOS
MDATE TABLE OVERFLOW	System software error; the internal loader table, MDATE, overflowed because the number of modules to load exceeded 2500. Notify a systems analyst.	8	LOAD
MESSAGE ERROR	Error in system GET A MESSAGE call.	8	LOAD

Table B-1. Diagnostic Messages (Sheet 13 of 24)

Message	Significance/Action	Return Code	Issued By
MISSING INPUT FILE	Correct control statement.	8	OLE
MODULE modulename HAS A ZERO LENGTH COMMON BLOCK blockname	Module is trying to define a zero length common block.	8	LOAD
MODULE LIMIT ON OBJECT FILE EXCEEDED	Notify a systems analyst.	8	OLE
MODULE modname INT DATA ** MODE = 1 TYPE ILLEGAL	Module modname has a mode 1 in an interpretive data type that is illegal; probably the most common with type 9 mode 1. Table type 101.	8	LOAD LOAD
MODULE modname INT REL ** MODE= 2 TYPE DOES NOT EXIST	Module modname has a mode 2 in an interpretive data type that does not exist. Table type 201.	8	LOAD
MODULE ON FILE filename HAD A MODULE HEADER LENGTH OF ZERO	Bad module header format in the input file.	8	OLE
MODULE ON FILE filename HAS NO HEADER	Module does not have a header table.	8	OLE
MODULE module-name NOT ON FILE filename	The module does not exist in the input file.	8	OLE
MODULES, NAMED, AND BLANK COMMON CANNOT BE GROUPED TOGETHER	The user specified blocks of more than one type (module, named common, and blank common) with a grouping parameter. Each type must be specified with a separate parameter.	8	LOAD
MORE THAN ONE ALTERNATE FILE WAS ATTEMPTED	Correct UPDATE directive.	8	UPDATE
MORE THAN ONE OMIT/SELECT FOR filename	Correct control statement.	8	OLE
MTuu message	See corresponding NTuu message.		
MULTIPLE TRANSFER SYMBOLS DEFINED	Two modules have been indicated as main programs; that is, both compilation units have a PROGRAM statement.	-	LOAD
NAME ALREADY EXISTS IN DIRECTORY	Change duplicate deck name.	8	UPDATE
NAME DOES NOT EXIST IN DIRECTORY	Change deck name or directive requested.	8	UPDATE

Table B-1. Diagnostic Messages (Sheet 14 of 24)

Message	Significance/Action	Return Code	Issued By
NEW PASSWORD MUST BE NON-BLANK	Password entry is required; therefore, a blank password is invalid. Correct the new password and try again.	8	PASSWORD
NO ALPHA POINTER	Virtual system was called for an Alpha message, but the pointer to the Alpha message was 0.	-	VSOS
NO DISC SPACE FOR EXTENSION ON FILE	Rerun job with this file on a disk that has enough space for this file.	-	VSOS
NO DROP FILE	There is no drop file for this task.	-	VSOS
NO ENTRY	No entry point found.	8	LOAD
NO ERROR EXIT ADDRESS	Task issued an Alpha/Beta system call which returned an error, but the error exit address field in the Alpha portion was 0.	-	VSOS
NO EXT/ENT POINTER FILE filename	No type 2 table pointer found.	8	LOAD
NO FILE	Either a file with the specified name does not exist, or the number of characters in filename was not one through eight.	-	VSOS
NO FILE NAME FOR READ	Add file name to READ directive.	8	UPDATE
NO FILES OR FST TABLE SPACE	System file index or file segment table full. Rerun.	-	VSOS
NO FILES TO LIST	No files exist that match the specifications on the FILES control statement.	-	FILES
NO FST ORDINAL WITH C50X REQUEST	Rerun with an FST ordinal.	-	VSOS
NO JOB CATEGORY FOUND FOR SPECIFIED LIMITS	The RESOURCE control statement specified values such that no job category can accommodate the job. Reduce the values and resubmit the job.	-	VSOS
NO LOGINS	Interactive terminal login lines are inhibited.	-	VSOS
NO MASS STORAGE FOR DROP FILE	Not enough mass storage available on the designated drop file packs. Rerun later.	-	VSOS

Table B-1. Diagnostic Messages (Sheet 15 of 24)

Message	Significance/Action	Return Code	Issued By
NO MESSAGE POINTER FOLLOWS EXIT FORCE	Rerun with a correct exit force.	-	VSOS
NO MORE SEGMENT SPACE IN FILEI FILE filename	The file has been extended three times and cannot be extended again.	-	VSOS
NO PARAMETERS SPECIFIED	Correct DEBUG control statement.	8	DEBUG
NO PP	No task in execution to break.	-	VSOS
NO SOURCE FILE	There is no controllee file.	-	VSOS
NO SUCH MODULE	Value entered for name=location was not the name of a module in the controllee.	4	DEBUG
NO SUCH SYMBOL	Symbol specified was not found as a symbol of currenttype (S or L) in current module.	4	DEBUG
NO SWITCH PARAMETERS FOUND, FILE NOT ALTERED.	File name was only parameter found.	4	SWITCH
NO TIME FOR USER	The user has no more time available. Either increase the time limit on the job RESOURCE statement or interactive task execute line or, if the maximum time limit is already specified, ask site personnel to increase the time limit for the user number.	0	VSOS
NO TIME LEFT FOR drop file filename/controller file filename	Rerun with a larger time limit. For batch jobs, 1/2 second of CPU time is added to allow for cleanup in exit processing.	-	VSOS BATCHPRO
NO TL	Zero time limit (TL) specified.	-	VSOS
NO WRITE PERMISSION - COMMAND IGNORED	Self-explanatory.	-	LOOK
NOID CANNOT BE WRITTEN TO NEWPL. UPDATE ABORT	Self-explanatory.	8	UPDATE
NONEXECUTABLE FILE	File requested is not a virtual code file (file type other than 2).	-	VSOS
NONMATCHING WORDS	Listed words did not compare equally.	4	COMPARE
NON-PRODUCTION PROGRAM NOT PERMITTED	A production user has attempted to execute a nonproduction program interactively. Contact the site security administrator.	-	VSOS
NOT A LOGIN	First word of the LOGIN command must be LOGIN.	-	VSOS

Table B-1. Diagnostic Messages (Sheet 16 of 24)

Message	Significance/Action	Return Code	Issued By
NOT A USER	Invalid user number.	-	VSOS
NOT ENOUGH TIME FOR THIS JOB	Time limit specified exceeds time remaining in repository bank.	-	VSOS
NOT EXPECTING SEQUENCE NUMBER	Correct UPDATE directive.	8	UPDATE
OLDPL DIRECTORY CANNOT BE PROCESSED	Notify a systems analyst.	8	UPDATE
OLD PASSWORD DOES NOT VERIFY	The old password specified does not match the password associated with the user number in its user directory entry. Correct the password and try again.	8	PASSWORD
OLE TERMINATED	Informative message.	-	OLE
OMIT PARAMETER MISSING A FILE OR MODULE	Correct control statement.	8	OLE
ON TTY xxxx	User numbers cannot logon to more than one terminal at a time.	-	VSOS
OPERATION NOT ALLOWED ON YANK\$\$\$	The YANK\$\$\$ deck cannot be purged. However, the user can delete directives within the YANK\$\$\$ deck.	8	OLE
OPERATOR NO.=uuuuuu	The system operator attempted to logon under an invalid user number. The correct operator number is uuuuuu.	-	VSOS
ORIGIN NOT ON LARGE PAGE BOUNDARY - ORIGIN = neworigin	Warning message; the GRLPALL option is specified so the origin address is changed to a large page boundary.	4	LOAD
OUT OF BOUND MEMORY REFERENCE	Attempt to access space outside the task virtual space.	-	VSOS
OUTADR EXCEEDS outfile FILE LENGTH	O parameter is greater than the number of words in outfile.	8	COPY
PACK packname IS NOT AVAILABLE	Try again when the device is available (UP).	8	DMAP
PAGE SIZE CONFLICT IN DROP FILE	Small page fault but the drop file is in large pages, or vice versa.	-	VSOS
PARAMETER OR FORMAT ERROR	Error in parameter specifications.	8	Pool Utilities

Table B-1. Diagnostic Messages (Sheet 17 of 24)

Message	Significance/Action	Return Code	Issued By
PARAMETER OR FORMAT ERROR FOUND. UTILITY TERMINATED	Correct FILES control statement.	8	FILES
PARAMETER OR FORMAT ERROR FOUND. UTILITY TERMINATED	Correct GIVE control statement. File remains with old owner.	8	GIVE
PARAMETER TOO LONG TO PROCESS	The parameter string contains a value or name that is too long.	8	LOAD
PC AND RMK MUST BE SINGLE CHARACTER OR HEX NUMBER FROM 0 TO FF	The padding character and/or the record mark character must be specified as the character or its hexadecimal code.	8	DEFINE REQUEST SWITCH
POOL pool ATTACHED	The system successfully attached the specified pool.	-	PATTACH
POOL pool CREATED	The system successfully created the specified pool.	-	PCREATE
POOL pool DESTROYED	The system successfully destroyed the specified pool.	-	PDESTROY
POOL pool DETACHED	The system successfully detached the specified pool.	-	PDETACH
POOL pool NOT FOUND	The specified pool does not exist. Correct the pool name and try again.		LISTAC
POOL FILE filename CURRENTLY OPENED	Self-explanatory.	4	PURGE
PREMATURE EOF ENCOUNTERED ON OLDPL. UPDATE ABORT	OLDPL has been truncated or damaged and must be rebuilt from the previous revision.	8	UPDATE
PROBLEM WITH FILE	Notify a systems analyst.	8	UPDATE
PROCESSED INVALID DIRECTIVE IN ALTERNATE FILE	Correct directive.	8	UPDATE
PURGE ERROR R=#rrr SS=#ss ON FILE filename	See DESTROY system message in volume 2 for an explanation of the error code.	8	PURGE
RANGE IS NOT PERMITTED FOR YANKDECK	Correct YANKDECK directive.	8	UPDATE
READ ERROR ON FILE	SIL detected an error. Refer to the SIL message.	8	UPDATE

Table B-1. Diagnostic Messages (Sheet 18 of 24)

Message	Significance/Action	Return Code	Issued By
READ RECOVERABLE ERRORS= nnnn	Informative message providing the number of read recoverable errors detected since the beginning of the tape volume. Additional detailed error information is recorded in the system error file accessible to site personnel.	-	-
RECURSIVE CALL NOT PERMITTED	Correct CALL references.	8	UPDATE
REQUESTED LP EXCEEDS INTRACTV LP LIMIT OF nnn PAGES	The large page limit specified on the execute line exceeds the maximum large page limit for the INTRACTV job category. Reenter the execute line specifying a large page limit not greater than nn pages.	8	VSOS
REQUESTED LP EXCEEDS JOB'S LP LIMIT OF nn PAGES	The user specified a large page limit that exceeds the maximum large page limit for the job. Correct the SET statement so the large page limit is not greater than nnn pages.	8	BATCHPRO
REQUESTED LP EXCEEDS TASK WS LIMIT OF nnnn BLOCKS	When multiplied by 128, the specified large page limit exceeds the working set size limit of nnnn blocks. Reenter the execute line specifying a larger working set size limit or a smaller large page limit.	8	XEQLN
REQUESTED WS EXCEEDS INTRACTV WS LIMIT OF nnnn BLOCKS	The user specified a working set size limit larger than the maximum working set size limit for the INTRACTV job category. Reenter the execute line specifying a working set size limit not greater than nnnn blocks.	8	XEQLN
REQUESTED WS EXCEEDS JOB'S WS LIMIT OF nnnn BLOCKS	The user specified a working set size limit that exceeds the maximum working set size limit for the job. Correct the SET statement so the working set size limit is not greater than nnnn blocks.	8	BATCHPRO
REQUESTED WS TOO SMALL FOR JOB'S LP LIMIT OF nnn PAGES	The user specified a working set limit smaller than the current large page field length. Correct the SET statement so the working set size limit is not smaller than nnn*128.	8	BATCHPRO

Table B-1. Diagnostic Messages (Sheet 19 of 24)

Message	Significance/Action	Return Code	Issued By
REQUIRED FILE NAME MISSING FOR POOL poolname	The user specified a pool name without specifying the file that belongs to the pool. Add a file name after the pool name.	8	PERMIT
REQUIRED PARAMETER MISSING. NEXT EXPRESSION IS: expression ENTER PARAMETER OR "CANCEL"	Required positional parameter missing. Given expression appeared in the position where a required parameter was expected.	8	Q7KEYWRD
REQUIRES USER DYN/SHARED LIB	Fatal message. A task is running that requires a user dynamic or shared library file and the files are not attached.	8	Q5INIT
RESPONSE CODE IS xxx	A fatal error occurred during interpretation of the MFLINK execute line. Response code xxx returned by Q7PROMPT.	8	MFLINK
RESTART OF DROPFILE NOT PERMITTED	A nonproduction user has attempted to interactively restart a drop file that has been flagged as nonrestartable by the system. Contact the site security administrator.	-	VSOS
SAY AGAIN	The special character (sc) is not followed by a valid interactive request character.	-	VSOS
SBU MEMORY PARITY ERROR	Parity error occurred on read or write.	-	VSOS
SECURITY LEVEL IGNORED	Warning message; the security level specified on the control statement is ignored because the specified file is a local file; it already has a security level.	4	DEFINE
SELECT PARAMETER MISSING A FILE OR MODULE	Correct control statement.	8	OLE
SEND AGAIN	The state at this DB entry is zero; or the job class is priority and the privileged job permission flag is zero; or the job is currently in interrupt mode and an explicit I/O interrupt has occurred.	-	VSOS
SEQUENCE NUMBER EXCEEDED	Create two decks if text cards exceed 65,535.	8	UPDATE
SEQUENCE NUMBER MISSING FROM DIRECTIVE	Add a sequence number to the directive.	8	UPDATE

Table B-1. Diagnostic Messages (Sheet 20 of 24)

Message	Significance/Action	Return Code	Issued By
SEQUENCE NUMBER NOT FOUND	Specified sequence number nonexistent. Correct.	8	UPDATE
SINGLE TRACK CORRECTABLE ERRORS=nnnn	Informative message providing the number of single-track correctable errors detected since the beginning of the the tape volume.	-	-
SOURCE OR DROP FILE ANOMALY	IOC in bound implicit map is not 16 (source); or bound implicit map entry is outside of file bounds; or drop file (free space) map address overlap occurred.	-	VSOS
SYNTAX ERROR ON DIRECTIVE	Correct error and resubmit.	8	UPDATE
SYSTEM DROP FILE CREATE ERROR	Either the disk or file index is full.	-	VSOS
SYSTEM INTERRUPTION, JOB NOT RERUN	Before the batch job completed, the system was interrupted and reauto-loaded. However, the job was not rerun either because a NORERUN control statement was processed or the operator prevented a job rerun. Resubmit the job.	-	VSOS
SYSTEM MESSAGE ERROR	Batch processor detected a system message error. Contents of Alpha and Beta words follow.	-	BATCHPRO
SYSTEM TABLES FULL, TRY AGAIN	The XEQ buffer table is full as more than eight execute lines have been entered; or no DB entry can be obtained; or no user table entries are available.	-	VSOS
TABLE TYPE NOT IMPLEMENTED	Refers to compiler output for the loader.	8	LOAD
TASK BROKEN; END OF JOB PROCESSING	BATCHPRO is terminating the job because the current task was killed. There is no EXIT card processing.	-	BATCHPRO
TASK BROKEN; EXIT CARD PROCESSING	BATCHPRO is advancing to the next EXIT card because the current task was dropped.	-	BATCHPRO
TASK SUSPENDED WAITING FOR PERMANENT FILE	A file specified on the ATTACH statement is already attached to another suffix. The user specified the WAIT parameter so the task is suspended until the file is available.	-	ATTACH

Table B-1. Diagnostic Messages (Sheet 21 of 24)

Message	Significance/Action	Return Code	Issued By
TEXT STRING EXCEEDS LIMIT	A string specified in the statement is longer than 2880 characters. Each string must be delimited by double quote (") characters.	8	Q7KEYWRD
TOO MANY ERRORS	Correct UPDATE directives.	8	UPDATE
TRANSMISSION PARITY ERROR	Parity error occurred on read or write.	-	VSOS
TROUBLE READING FILE - xxxxxxx	Notify a systems analyst.	8	UPDATE
TRY AGAIN	ROLL, BACK, a display or a display register without parameters was given before a DISPLAY or ENTRY command and DEBUG has no point of reference for ROLL/BACK. CONTINUE was given before EXECUTE. A second EXECUTE command is given. STEP is given before EXECUTE.	4	DEBUG
UNABLE TO DESTROY POOL	The pool cannot be destroyed for one of the following reasons: <ul style="list-style-type: none"> • The pool is attached to another job. • Files exist that belong to the pool. • The user is not the pool boss. 	4	PDESTROY
UNABLE TO ENLARGE DROP FILE. TRY AGAIN	Self-explanatory.	8	PURGE
UNABLE TO EXPAND FOLLOWING COMDECK - xxxxxxx	Specified COMDECK is missing and should be supplied.	4	UPDATE
UNABLE TO FIND EXECUTE FILE	The system cannot access the execute file for the task being restarted.	-	VSOS
UNABLE TO FIND EXTERNAL-ENTRY TABLE FOR MODULE modname ON FILE filename	OLE could not find the external reference table dueto bad module structure.	8	OLE
UNABLE TO GET TIME AND DATE	Notify a systems analyst.	8	OLE
UNDEFINED NAME OR ALREADY IN GROUP	Grouping not done because of an undefined name or the element to be grouped is already in another group.	8	LOAD

Table B-1. Diagnostic Messages (Sheet 22 of 24)

Message	Significance/Action	Return Code	Issued By
UNEXPECTED ERROR ON VALIDATION OF ACCOUNT	System table incompatibility. Notify a systems analyst.	-	VSOS
UPDATE OBTAINED BAD DATA IN PROCESSING THIS CARD	Correct card.	8	UPDATE
USER LOCKED OUT OF SPECIFIED JOB CATEGORY	The user is not validated to use the job category specified on the RESOURCE statement. Specify another mnemonic such as the default, JDEFAULT.	8	VSOS
USER NOT VALIDATED FOR TAPE USAGE	The RESOURCE control statement on a job requested a tape with the NT parameter but the user is not validated for tape access.	8	VSOS
USER usernum NOT FOUND LISTAC	The specified user number does not exist. Correct the user number and try again.		
VALIDATE OPTION - MUST BE Y/N	Option specified must be Y or N.	8	LOAD
VARIABLE RATES NOT DEFINED AT THIS INSTALLATION	The control statement specified the VRI parameter, but the site did not install variable rate accounting (system installation parameter IP_F_VR is zero).	8	EDITPUB
WARNING *** ATTACHED POOLS	Informative message during LOGON. The user has attached pools.	-	VSOS
WARNING - ATTRIBUTES OF POOL FILE MAY NOT MATCH INPUT FILE	The COPY utility cannot change pool file attributes.	4	COPY
WARNING - DISK TAPE PARAMETERS IGNORED FOR TAPE FILE DISK	The REQUEST utility will ignore inappropriate combinations of file attribute parameters.	4	REQUEST
WARNING * DUPLICATE FILES	Informative message during LOGON. The user has duplicate files.	-	VSOS
WARNING - INPUT FILE IS EMPTY, NO DATA TO COPY	Informative message; COPY copied no data.	4	COPY
WARNING MODULE name FROM FILE filename IS INACCESSIBLE AND THEREFORE NOT LOADED	Module is deleted from controllee since it cannot be referenced.	4	LOAD

Table B-1. Diagnostic Messages (Sheet 23 of 24)

Message	Significance/Action	Return Code	Issued By
WARNING MULTIPLE TRANSFER SYMBOLS DEFINED	More than one program entry point is defined.	4	LOAD
WARNING - NO OBJECT FILE CREATED	Informative message.	4	OLE
WARNING - OBJECT FILE LENGTH INCREASED TO nnnn PAGES	Informative message.	4	OLE
WARNING - PACKID IGNORED. filename ON PACK packid	File filename is not on the pack specified on the COPY statement. It is on pack packid and the file was copied to that pack.	4	COPY
WARNING - 'POOL=' PARAM NOT SPECIFIED- 'PRIVI=' PARAM IGNORED	If the POOL parameter is omitted, the PRIVILEGED parameter is ignored.	4	GIVE
WARNING UNSATISFIED EXTERNAL(S) DETECTED DURING LOAD	Routine referenced but not provided.	4	LOAD
WARNING xxxxxxxx IS DUPLICATE ENTRY POINT IN MODULES yyyyyyyy AND zzzzzzzz	xxxxxxx is the entry name, and yyyyyyy and zzzzzzz are the module names. References to xxxxxxxx link to the entry in yyyyyyy, which was the first encountered.	4	LOAD
WIDTH EXCEEDS 256 CHARACTERS	Line image width on *WIDTH directive exceeds 256 characters.	4	UPDATE
nnnnnn WORDS OF FILE infile COPIED TO FILE outfile	The system copied nnnnnn words from file infile to file outfile.	-	COPY
taskname WORKING SET TOO SMALL nn% OF CPU TIME	The maximum working set size for task taskname was too small for nn% of its execution time. The user should consider increasing the maximum working set size limit to decrease the paging required for the task.	4	BATCHPRO
WIDTH EXCEEDS 256 CHARACTERS	Correct *WIDTH directive.	4	UPDATE
WRITE ON READ-ONLY FILE	Job attempted to write to temporary space for which the job does not have write access.	-	VSOS

Table B-1. Diagnostic Messages (Sheet 24 of 24)

Message	Significance/Action	Return Code	Issued By
WRITE RECOVERABLE ERRORS=nnnn	Informative message providing the number of write recoverable errors detected since the beginning of the tape volume. Additional detailed error information is recorded in the system error file accessible to site personnel.	-	-
WRITE VIOLATION IN SYSTEM CALL	Operating system error on read-only file.	-	VSOS
WRONG TARGET PAGE SIZE	The TSP parameter specified a small page size other than one, four, or sixteen blocks. Correct the TSP parameter and try again.	8	LOAD

Table B-2.. System Utility Error Messages (Sheet 1 of 80)

Sever-ity	Error Code	Message	Significance	Issuing Routine
F	0001 to 0199	ILLEGAL PARAMETER parameter	The user specified an invalid or incorrect parameter. The status code in the message is the ordinal of the parameter within the parameter sequence.	ALL SIL
F	0001	NO MATCH FOR ERROR CODE IN ECODE	Utility error; notify a systems analyst.	PFERROR
F	0002	UNEXPECTED PROGRAM ERROR	Utility error; notify a systems analyst.	DUMPF GENLINE PFRHDIO PFRHLIO
F	0003	NO FILES TO xxxxxx	No files exist that match the qualifications on the statement so no files are processed (audited, loaded, or dumped).	AUDIT DUMPF LOADPF
F	0004	MASTER DIRECTORY FILE filename IS FULL	The master directory file for the current disk pack is full. A master directory file is created on the next disk pack listed on the VSN parameter and dumping continues on that pack.	DUMPF
F	0005	THERE ARE TOO MANY FILES TO PROCESS	The statement specifies more files than can be processed at one time. Decrease the number of files.	DUMPF PFGETNFI
F	0006	EXCEEDED SPECIFIED PACK NAMES	Space on the specified packs is insufficient for the files; additional space is used on the default system pack.	PFBUFFER
F	0008	INVALID JCS	The job control string specified on the JCS parameter is invalid. Ask a systems analyst for the correct job control string.	PFSCAN
F	0009	TOO MANY JCS SPECIFIED	The JCS parameter specified more than 10 strings. Reduce the number of strings specified on the parameter.	PFSCAN

Table B-2. System Utility Error Messages (Sheet 2 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	0010	VSNS ARE REQUIRED	The control statement must specify the volume serial numbers on the VSN parameters. Specify the VSN parameter.	DUMPF
F	0011	INVALID SI setid	The set identifier specified on the SI parameter is invalid. Ask a systems analyst for the correct set identifier.	PFSCAN
F	0012	ATTEMPTED PRIVILEGED archive/audit BY NON-PRIVILEGED USER	An attempt was made to perform functions that require the user to be privileged.	AUDIT DUMPF LOADPF PFSCAN
F	0013	RHF NOT AVAILABLE	RHF parameters were specified on the control statement of a version of DUMPF or LOADPF that does not have RHF capabilities implemented.	AUDIT DUMPF LOADPF
F	0014	TOO MANY PACKS IN DST	There are too many packs configured in the system for the utility to run. Consult a systems analyst.	PFSCAN
F	0016	UNEXPECTED SYSTEM SS CODE sscode FROM FC=fc	An unexpected error occurred when a system message was issued. Consult a systems analyst.	PFUSER

Table B-2. System Utility Error Messages (Sheet 3 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
W	0017	UNABLE TO GIVE qfile TO INPUT QUEUE - RCODE nn	<p>An error occurred when LOADPF attempted to reload an archived input file. The reason is given by nn, which has one of these values:</p> <ol style="list-style-type: none"> 1 Job category does not exist. 2 Maximum working set limit exceeded. 3 Large page limit exceeded. 4 Invalid time limit. 5 Invalid priority. 6 Large page limit exceeds maximum working set. 7 Invalid user number. 8 Invalid resource card. 9 Invalid resource card parameter. 10 User locked out of specified job category. 11 Input queue is full. 12 Unexpected error in entering job to input queue. 13 Invalid user, account, password, or security level. 14 Maximum permissible number of jobs exceeded. 15 Maximum large page limit exceeded for specified job category. 16 No job category found for specified limits. 17 No tape access allowed for user. 18 Input queue limit for user is exceeded. 19 No jdns available to assign to input file. 	LOADPF

Table B-2. System Utility Error Messages (Sheet 4 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	0018	TRANSFER OF FILE file REJECTED BY REMOTE HOST	The remote host reject the file transfer. Consult job dayfile for specific reason.	PFRHLIO
F	0019	ILLEGAL PARAMETER	The user specified an illegal parameter in the control statement.	PFSCAN
F	0020	VSN xxxxxxxx IS INCORRECT	The format of one of the VSNs specified is invalid. Device set VSNs must follow the same convention as file names. Tape VSNs can consist of any combination of alphanumeric characters but can only be one to six characters long.	PFSCAN
F	0021	INVALID ACCOUNT IDENTIFIER - xxxxxxxx	The user specified an invalid account identifier in the control statement. xxxxxxxx is the account identifier in error.	AUDIT
F	0022	NOT A MASTER USER OF SPECIFIED ACCOUNT	The user is not a master user for the specified account.	AUDIT
F	0023	USER NOT ALLOWED TO USE SELECT VALUE param	A user attempted to execute an AUDIT/DUMPF/LOADPF utility with the SELECT parameter set to I, O, or IO on a user number other than the system user number.	AUDIT DUMPF LOADPF
W	0024	SPECIFIED FILE lfn NOT FOUND	A user specified a file on the lfn-list parameter of the AUDIT/DUMPF/LOADPF utility that could not be located.	AUDIT DUMPF LOADPF
F	0025	FILE lfn ON USER usernum HAS A BAD FILEI	The file lfn on user number usernum cannot be processed because of a bad FILEI associated with the file. Consult a systems analyst.	AUDIT DUMPF LOADPF
F	0200	SIL BUG-ILLEGAL OPTION	SIL specified an illegal option for a system message. Consult a systems analyst.	ALL SIL

Table B-2. System Utility Error Messages (Sheet 5 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	0201	SIL BUG-ILLEGAL BETA	The Beta area SIL supplied for the system message was rejected by the system. Consult a systems analyst.	ALL SIL
F	0202	SIL BUG-UNRECOGNIZED ERROR CODE	Unrecognized R code in an Alpha or an unrecognized SS code in a Beta. Consult a systems analyst.	ALL SIL
F	0203	INTERNAL CALL TO routine FAILED, STATUS status	Internal error. Consult a systems analyst.	ALL SIL
F	0204	SIL BUG - UNEXPECTED R CODE code FROM FC code	Internal error. Consult a systems analyst.	ALL SIL
F	0205	SIL BUG - UNEXPECTED SS CODE code FROM FC code	Internal error. Consult a systems analyst.	ALL SIL
F	0206	SIL BUG - UNEXPECTED CERR CODE code FROM FC code	Internal error. Consult a systems analyst.	ALL SIL
F	0207	SIL BUG - UNEXPECTED SERR CODE code FROM FC code	Internal error. Consult a systems analyst.	ALL SIL
F	0210	NO MATCH FOR SYSERR IN R_VCODE TABLE	The SYSERR code passed to the internal routine Q5_PERR is not in the T_RCODES table. Consult a systems analyst.	ALL SIL
F	0211	NO MATCH FOR VCODE IN T_MVCT	A VCODE found in the T_RCODES table is not in the T_MVCT table within internal routine Q5_PERR. Consult a systems analyst.	ALL SIL
F	0250	REQUIRED PARAMETER parameter MISSING	The user omitted a required parameter. Refer to the call description.	ALL SIL

Table B-2. System Utility Error Messages (Sheet 6 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	0250	REQUIRED PARAMETER parameter MISSING	You omitted a required parameter. Refer to the control statement description in chapter 6.	LOADPF PFSCAN
F	0251	DUPLICATE FILE NAME name	The user specified a file name which already has a FIT.	SIL SIL
F	0252	REQUIRED PARAMETER FOR SET n MISSING	The user did not specify a required parameter identifying the file or the action to be performed by the call.	ALL SIL
F	0253	INVALID LFN filename	SIL did not recognize the specified file name.	ALL SIL
F	0254	INVALID FLUN rflun	A FIT does not currently exist having the specified file logical unit number. To obtain a file's flun, specify the RFLUN= parameter on the Q5DEFINE, Q5GENFIT, or Q5RQUEST call that generates the file's FIT.	ALL SIL
F	0255	MUTUALLY EXCLUSIVE PARAMETER SPECIFIED FOR FILE filename	The user specified two or more mutually exclusive parameters. Refer to the call description to determine which parameter to omit.	ALL SIL
F	0255	MUTUALLY EXCLUSIVE PARMETERS	The control statement cannot specify both the JCS and INPUT parameters or the U option with any other parameter. Omit one of the parameters.	PFSCAN
F	0256	ALL ALPHAS IN USE, CALL Q5CHECK	The maximum allowed concurrent I/O requests are pending. Call Q5CHECK to determine whether an I/O request has completed.	Q5READ Q5WRITE
W	0257	CONTROLLEE(S) DON'T MATCH CURRENT SYSTEM LEVEL	One or more controllee files were loaded but were generated at a different operating system level than the current one. These files may not execute correctly. Recompile and/or regeneration of the controllee files (using the LOAD statement) may be necessary.	LOADPF
F	0258	FILE filename HAS PURGE ONLY ACCESS	Permanent file validation has found that the named file is not usable.	Q5OPEN Q5ATTACH Q5INIT Q5INITCH

Table B-2. System Utility Error Messages (Sheet 7 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	0261	TOO MANY PARAMETERS	The user specified more than 199 parameters on the call.	ALL SIL Q5SNMJS
F	0262	ILLEGAL MNEMONIC FOR PARAMETER	The user specified an invalid mnemonic value on aa parameter. Correct the mnemonic and try again.	Q5ATTACH Q5DCDPFI Q5LFIPOL Q5LFIPRI Q5LFIPUB
W	0263	ILLEGAL PARAMETERS FOR CONNECTED FILE filename IGNORED	One or more parameters specified on the call are ignored because the file name specified on the call, filename, is that of a file connected to a terminal. Refer to the call description for the valid parameters.	Q5DCDPFI
F	0264	MUTUALLY EXCLUSIVE PARAMETER SPECIFIED	The user has specified two parameters that cannot be used at the same time.	Q5GETPFI Q5DCDPFI Q5LFIPOL Q5LFIPRI Q5LFIPUB Q5PERMIT
F	0265	USER NOT PRIVILEGED OR INVALID USER NUMBER	The user number executing the LIMITS command is not privileged or has specified an invalid user number.	LIMITS
W	0300	MORE FILES TO LIST	More file indices exist than can fit in the buffer area SIL defines. Consult a systems analyst.	Q5GETPFI Q5LFIPOL Q5LFIPRI Q5LFIPUB
F	0301	POOL poolname DOES NOT EXIST OR IS NOT ATTACHED	The specified pool either does not exist or is not attached. Check that the correct pool is specified on the call.	Q5LFIPOL Q5GENFIT Q5PATACH Q5PDESTR Q5PDTACH Q5PGRACC Q5PREACC Q5PERMIT Q5USERL
F	0302	PERMANENT FILE filename WAS NOT FOUND	The specified permanent file does not exist or the user does not have access to the file.	Q5ATTACH
W	0303	INVALID FILENAME filename	The syntax of the specified file name is incorrect. A file name is a sequence of one to eight characters beginning with a letter.	ALL

Table B-2. System Utility Error Messages (Sheet 8 of 80)

Sever-ity	Error Code	Message	Significance	Issuing Routine
W	0304	NO FILES QUALIFY	No files match the qualifiers specified on the call.	Q5LFIPOL Q5LFIPRI Q5LFIPUB Q5PERMIT
F	0306	USER TABLE IS FULL	The system user activity table is full. Notify a systems analyst and try to rerun job.	ALL SIL
F	0307	FILEI IS FULL	The system file index is full. Notify a systems analyst, and try to rerun job.	ALL SIL
F	0308	INVALID USER NUMBER	The user number specified is syntactically incorrect, or does not exist.	ALL SIL
F	0309	NOT A MASTER USER OR PRIVILEGED USER	The user is not the master user of the account identifier, or the user is not privileged.	Q5GETPFI
F	0310	NONPRIVILEGED USER	A nonprivileged user attempted to issue a privileged call.	Q5GETPFI Q5PERMIT Q5SNDMJS Q5CHANGE
F	0311	DISK I/O ERROR ON PACK packid	An error was encountered when attempting to read the PFI of the specified disk.	Q5GETPFI
F	0312	packid PACKID NOT FOUND	The specified disk is not currently available.	Q5GETPFI
F	0312	device set DEVICE SET NOT FOUND	Either the specified device set is not available or it is not in the disk status table.	AUDIT DUMPF PFSCAN
W	0313	DROP FILE xxxxxxxx HAS BEEN RENAMED yyyyyyy	Nonprivileged users cannot reload archived drop files unless they are renamed using a name that doesn't begin with a number. Thus, to reload drop file xxxxxxxx, it was renamed yyyyyyy.	PFLIO
W	0314	ILLEGAL DATE xxxxxxxx CORRECTED TO yyyyyyy	The date xxxxxxxx specified by the DATE keyword is invalid. The utility changed the date to yyyyyyy. Both dates have the format mddy (month, day, year).	PFDATEGJ

Table B-2. System Utility Error Messages (Sheet 9 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	0320	ILLEGAL MESSAGE LENGTH	The message length is zero.	Q5SNDMCR Q5SNDMCE Q5SNDMDF Q5SNDMJC Q5SNDMOP
F	0321	ILLEGAL DESTINATION	The call specified a controller or controllee that does not exist.	Q5MSGCTR Q5SNDMCE Q5SNDMCR Q5SNDMJC Q5SNDSTR
F	0322	LOGGED OUT TERMINAL	The controller specified in the request is a logged out terminal.	Q5SNDMCR
F	0324	SYSTEM BUFFER BUSY	The system buffer is busy. Try again later.	Q5SNDMCR Q5SNDMJC Q5SNDMJS
F	0325	CONTROLLEE BUSY	The controllee which is the destination of the message already has text from a controller.	Q5SNDMCE
F	0326	NO OPERATOR COMMUNICATION	Either the operator is not logged on or the system buffer is full. If the user specified the SAVE parameter on the call, the message is stored in the save table for later access by the operator.	Q5SNDMOP
W	0327	DAYFILE FULL, MESSAGES WILL BE LOST	The user's job dayfile is full. A warning has been entered in the dayfile. Messages will be overwritten for the duration of the job.	Q5SNDMDF
F	0328	DAYFILE NOT OPEN	The dayfile is not open for implicit I/O. Consult a systems analyst.	Q5SNDMDF
F	0329	ILLEGAL VBA FOR DAYFILE	Illegal virtual byte address for dayfile.	Q5SNDMDF
F	0330	DAYFILE NOT FOUND	The system cannot find the Q5DAYFLE file. Q5SNDMDF cannot be called from an interactive task. If the call was from a batch job, notify a systems analyst.	Q5SNDMDF
F	0337	ERROR IN SENDING MESSAGE TO DAYFILE	The batch processor received a message that it could not put in the dayfile.	Q5SNDMCR Q5SNDMJC

Table B-2. System Utility Error Messages (Sheet 10 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	0339	PARAMETER NULLFILL OR RJUSTIFY SPECIFIED WITHOUT STD OR SYD	The user specified the NULLFILL or RJUSTIFY parameter without specifying the STD or SYD parameter.	Q5GETMCE
F	0340	ILLEGAL BUFFER LENGTH	The user's buffer is too short or too long.	Q5GETMPG
F	0341	NO MESSAGE AVAILABLE	No message is waiting for this task.	Q5GETMCR Q5GETMOP Q5GETMCE
F	0342	ILLEGAL MESSAGE LENGTH	SIL encountered more than 200 delimiters in the message.	Q5GETMCE Q5GETMCR Q5GETMOP
F	0343	LEVEL 1 TASK	The caller is a level 1 task and therefore cannot have a controller from which to obtain a message.	Q5GETMCR
F	0344	CONTROLLER MESSAGE WAITING	A message cannot be transmitted because the task has a message from a controller waiting.	Q5GETMCE
F	0345	CONTROLLEE WAITING FOR MESSAGE	The controllee from which a message is expected is waiting.	Q5GETMCE
F	0346	BAD MINUS PAGE	When initializing the controllee, SIL found an error in a minus page value. Discard controllee file and generate a new controllee file with the LOAD utility.	Q5INITCH
F	0347	NO FST SPACE	System software error; no space is available for entries in the file segment table. Notify a systems analyst.	Q5INITCH Q5OPEN
F	0349	IOC FILE NOT FOUND	A file specified in the IOC cannot be found in the file index table (FILEI). Ensure that all files required by the program exist.	Q5INITCH
F	0350	CONTROLLEE ALREADY EXISTS	The controllee cannot be initiated because a controllee already exists.	Q5INIT Q5INITCH
F	0351	CONTROLLEE filename NOT FOUND	The controllee file filename does not exist.	Q5INIT Q5INITCH

Table B-2. System Utility Error Messages (Sheet 11 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	0352	NOT ENOUGH TIME	There is insufficient time to run the controllee.	Q5INIT Q5INITCH
F	0353	ILLEGAL PRIORITY	An illegal priority value was specified.	Q5INIT Q5INITCH
F	0354	DROPPFILE CREATE ERROR	An error was encountered when attempting to create the drop file.	Q5INIT
F	0355	filename NOT EXECUTABLE	The controllee program file filename is not executable.	Q5INIT Q5INITCH
F	0356	filename IO ERROR	A mass storage error was encountered when attempting to read the controllee program file filename.	Q5INIT Q5INITCH
F	0357	SYSTEM TABLES FULL	Controllee cannot be initiated because the system tables are full. Try again later.	Q5INIT Q5INITCH
F	0358	filename ABNORMALITY	The controllee file filename cannot be initiated because of an abnormality in the file or in the drop file I/O number.	Q5INIT Q5INITCH
F	0359	TOO MANY LEVELS	Controllee cannot be initiated because the controllee chain already contains nine tasks.	Q5INIT Q5INITCH
F	0360	DROPPFILE TOO SMALL	Controllee cannot be initiated because the drop file is too small.	Q5INIT Q5INITCH
F	0361	PERSISTENT DROPPFILE	System unable to destroy existing dropfile.	Q5INIT Q5INITCH
F	0362	INTERRUPT TABLE FULL	Controllee cannot be restarted because the interrupt register table is full. Try again later.	Q5INIT
F	0363	DROPPFILE VERIFY ERROR	Controllee cannot be initiated because the drop file cannot be verified.	Q5INIT
F	0364	DISK READ BUFFER FULL	Insufficient system buffer space to initiate task. Try again later.	Q5INIT
F	0365	filename BAD MINUS PAGE	The controllee file (filename) contains a bad minus page.	Q5INIT

Table B-2. System Utility Error Messages (Sheet 12 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	0366	SYSTEM BUG-DROPPFILE VERIFICATION	System detected an undefined error in drop file verification. System error.	Q5INIT
F	0367	FILE filename OPENED IN A PRIVILEGED MODE	Controllee program file is currently open (using a privileged OPEN) to a privileged user.	Q5INIT Q5INITCH
F	0368	SOURCE FILE BAD SMALL PAGE SIZE	The controllee cannot be initiated due to bad source file page size.	Q5INIT Q5INITCH
F	0369	DROP FILE BAD SMALL PAGE SIZE	The controller cannot be initiated due to bad drop file small page size.	Q5INIT Q5INITCH
F	0370	NO CONTROLLEE TO DISCONNECT	No controllee exists to disconnect.	Q5TERMCE
W	0371	EMPTY MESSAGE RECORD	An interactive user entered an EOR interactive request line in response to an input prompt thereby sending an empty record delimiter to the task.	Q5GETMCR
W	0372	EMPTY MESSAGE GROUP	An interactive user entered an EOG interactive request line in response to an input prompt thereby sending a group delimiter to the task.	Q5GETMCR
W	0373	EMPTY MESSAGE FILE	An interactive user entered an EOR interactive request line in response to an input prompt thereby sending a file delimiter to the task.	Q5GETMCR
F	0374	CHARGE STATEMENT MUST BE SUPPLIED	The user must execute a CHARGE statement.	Q5INIT
F	0375	EXCESSIVE CALLS	The task attempted to call the Q5CPUTIM routine more times than allowed by the installation parameter setting.	Q5CPUTIM
F	0376	INTERNAL ERROR - USER DIRECTORY NOT FOUND	The user directory was not found for an active user. Notify a systems analyst.	Q5GETPFI

Table B-2. System Utility Error Messages (Sheet 13 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	0378	USER REPRIEVE ENABLE OR DISABLE NOT ALLOWED	The request to enable or disable user reprieve processing failed. A site accounting routine could have prevented the reprieve request.	Q5REPREV
F	0380	INTERRUPT ADDRESS ERROR	The program interrupt address is the greater than virtual address range.	Q5DISAMI Q5ENAMI
F	0381	INTERRUPT OR DATA BASE ADDRESS ERROR	The program interrupt address or the data base address is greater than the virtual address range.	Q5ENATI Q5REPREV Q5RFI
F	0382	DATA BASE LENGTH OUT OF RANGE	The user specified a data base length that is out of range.	Q5ENATI Q5REPREV
F	0383	USER NOT IN INTERRUPT MODE	The user attempted to get interrupted program information when the program had not been interrupted.	Q5GETIIP Q5GETIRF
F	0384	DROP FILE TOO LONG	The controllee drop file is too long.	Q5INITCH
F	0385	FILE IS INCOMPLETE	The controllee file is incomplete.	Q5INIT Q5INITCH
F	0386	UNEXPECTED ERROR ON VALIDATION OF ACCOUNT	System table incompatibility. Notify systems analyst.	Q5INIT
F	0387	POOL poolname ACCESS DIRECTORY FULL	The specified pool access directory is full.	Q5PGRACC
F	0390	REQUESTED LARGE PAGE LIMIT EXCEEDS MAX WORKING SET LIMIT	The number of blocks specified as the new large page limit (LP * 128) is greater than the the number of blocks allowed for the task working set.	Q5SETLP
W	0391	REQUESTED LARGE PAGE LIMIT EXCEEDS MAX LARGE PAGE LIMIT	The current large page limit specified on the Q5SETLP call exceeded the maximum large page limit for the task. Therefore, the current large page limit was set to the maximum large page limit.	Q5SETLP
F	0400	filename DOES NOT EXIST	The specified batch file filename does not exist.	Q5DESBIF Q5RUNBIF
F	0410	ILLEGAL MESSAGE	The program cannot issue the Recall system message.	Q5RECALL

Table B-2. System Utility Error Messages (Sheet 14 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	0420	NON INTERRUPT ROUTINE	The program issuing the Q5RFI call is not an interrupt subroutine.	Q5RFI
F	0430	USER IS NOT PRIVILEGED OR THE OPERATOR	A nonprivileged user attempted to close a file opened for a privileged user.	Q5CLOSE
F	0434	UNDEFINED USER NUMBER	An invalid user number has been specified.	Q5GRACC Q5PREACC
F	0450	ADDRESS OUT OF USER VIRTUAL RANGE	The caller specified an address outside the user's virtual address range with the OUTADDR=, OUTDESC=, INADDR=, or INDESC= parameter.	Q5ADVISE
F	0451	ILLEGAL LENGTH	The length specified by the user was either too long or zero.	Q5ADVISE
W	0452	ADVISE IN	Only part of the virtual space requested was paged in because of insufficient memory to accommodate the entire specified virtual address range.	Q5ADVISE
W	0453	PARTIAL ADVISE OUT	Only part of the virtual space specified as no longer needed was paged out because a page within the specified range was locked down.	Q5ADVISE
W	0454	PARTIAL ADVISE REPLACE	Only part of the virtual space replacement was performed. The reasons are given under errors 0452 and 0453.	Q5ADVISE
W	0455	PAGE ALREADY IN CORE	A page of the virtual space requested paged in was already in memory. The rest of the space is paged in.	Q5ADVISE
W	0456	PARTIAL ADVISE - 0452 AND 0455	Only part of the requested space was paged due to errors 0452 and 0455.	Q5ADVISE
W	0457	PARTIAL ADVISE - 0453 AND 0455	Only part of the requested space was paged due to errors 0453 and 0455.	Q5ADVISE

Table B-2. System Utility Error Messages (Sheet 15 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
W	0458	PARTIAL ADVISE - 0452, 0453, AND 0455	Only part of the request space was paged due to errors 0452, 0453, and 0455.	Q5ADVISE
F	0459	ADVISE OUT ADDRESS MISSING	The length of the space to be paged out was specified, but not its address.	Q5ADVISE
F	0460	ADVISE IN ADDRESS MISSING	The length of the space to be paged in was specified, but not its address.	Q5ADVISE
F	0461	ADVISE OUT LENGTH NOT POSITIVE	A positive value must be specified as the length of the area to be paged out.	Q5ADVISE
F	0462	ADVISE IN LENGTH NOT POSITIVE	A positive value must be specified as the length of the area to be paged in.	Q5ADVISE
F	0463	ADVISE OUT PAGE COUNT TOO LARGE	The length of the space to be paged out is too large to fit in the appropriate field of the system message.	Q5ADVISE
F	0464	ADVISE IN PAGE COUNT TOO LARGE	The length of the space to be paged in is too large to fit in the appropriate field of the system message.	Q5ADVISE
F	0470	CALL NOT VALID FROM THIS TASK	The task cannot issue a Q5VRACC call because it is not public or its variable rate permit flag is not set.	Q5VRACC
F	0471	VARIABLE RATE ACCOUNTING NOT VALID AT THIS INSTALLATION	The site has set an installation parameter preventing use of variable rate accounting.	Q5VRACC
F	0472	CALL NOT VALID ON THIS SYSTEM	The site did not install variable rate accounting on the system.	Q5VRACC
F	0473	ACCOUNTING ACCESS DENIED	The task cannot change its accounting rate. Either the Q5VRACC call specified the wrong password or the user number is not validated for variable rate changes or the call failed a site-specified test.	Q5VRACC
F	0480	NO CONTROLLEE TO LIST	No controllee was found.	Q5GETCEN

Table B-2. System Utility Error Messages (Sheet 16 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
W	0481	MORE LEVELS TO LIST	More tasks are in the chain than the number the user specified on the NLVL= parameter.	Q5LSTCH
F	0485	NO CONTROLLER TO LIST	No controller was found. This message can be received by an interactive task, but not by a batch job.	Q5GETCRN
F	0504	ILLEGAL PFI ORDINAL	The ENTRY= parameter is either zero or greater than the number of entries in the SIL-defined area.	Q5DCDPFI
F	0505	ILLEGAL ROUTINE USAGE	The user did not call the routine (Q5GETPFI, Q5LFIPRI, Q5LFIPOL, or Q5LFIPUB) before issuing this call and did not specify the MYFILE= parameter on this call.	Q5DCDPFI Q5DCDPLB
F	0506	STLEN WRONG SIZE	An incorrect length was specified for the file segment table.	Q5DCDPFI
W	0507	KWD'S STLEN = AND * WILL BE IGNORED	The parameters STLEN=, SEGLEN=, and SEGADR= of Q5DCDPFI, if specified by the user, are ignored in a VSOS 2.2 and later operating system. The * is filled in by either a SEGLEN= or a SEGADR=, whichever was specified.	Q5DCDPFI
F	0508	MYFILE WRONG SIZE	An incorrect length was specified by the MYLEN= parameter for the file index entry length.	Q5DCDPFI
F	0509	JOB SESSION NOT ACTIVE	The calling task attempted to send a message to a job session that was no longer active.	Q5SNDMJS
F	0510	NONPRIVILEGED USER USED PRIVILEGED PARAMETER	A nonprivileged user attempted to use a privileged parameter.	Q5CHANGE Q5GIVE
W	0511	MESSAGE TRUNCATED TO 2000 CHARACTERS	The message contained in the MSG array was greater than 2000 characters. Only 2000 characters were sent to the job session.	Q5SNDMJS
F	0512	CUSER xxxxxxxx IS NOT A VALID USER NUMBER	The user number specified in the CUSER parameter is not a valid user number.	Q5GIVE

Table B-2. System Utility Error Messages (Sheet 17 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	0513	UNABLE TO GIVE ATTACHED PERMANENT FILE filename	The calling task used the CUSER= parameter but tried to give a file that was currently attached.	Q5GIVE
F	0516	NON-PRODUCTION PROGRAM NOT PERMITTED	A production user number has attempted to execute a nonproduction program. Contact the site security administrator.	Q5INIT Q5INITCH
F	0517	CALLER NOT SITE ADMINISTRATOR	Caller has attempted to use parameters or options reserved for user by the site security administrator.	Q5PERMIT Q5CHANGE
F	0518	WRITE PERMISSIONS NOT VALID FOR PRODUCTION CONTROLLEE	Write permissions cannot be given to a production controllee file. Contact the site security administrator.	Q5PERMIT
F	0519	FILE IS NOT A DROPFILE	Caller has attempted to enable the restart of a file that is not a drop file.	Q5CHANGE
F	0520	RESTART OF DROPFILE NOT PERMITTED	An attempt has been made to restart a drop file that has been flagged by the system as non-restartable. Contact the site security administrator.	Q5INITCH
W	0521	PRODUCTION STATUS LOST ON FILE filename	Privileged user number has reloaded a production file. Contact the site security administrator to reestablish its production status.	Q5DEFINE
W	0522	WRITE PERMISSIONS NOT VALID FOR DROPFILE	Write permissions cannot be given to a drop file and are ignored if specified.	Q5PERMIT
W	0523	ZIPCODE * IS NOT FOR A LOGGED ON WORKSTATION	The user tried to run a workstation utility and the workstation was not logged in.	Q5DCMSC
F	1400	SIX I/O REQUESTS STILL PENDING FOR FILE filename	The user cannot issue another file I/O request for the specified until one of its outstanding file requests completes. The user should issue a Q5CHECK call specifying that SIL wait until the request completes before returning control to the caller.	Q5ENDPAR Q5READ Q5WRITE

Table B-2. System Utility Error Messages (Sheet 18 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1401	NO I/O BUFFER SPECIFIED FOR FILE filename	The user did not specify an I/O buffer in the file's FIT or on the I/O request.	Q5GETN Q5GETP Q5PUTN Q5PUTP Q5READ Q5SKIP Q5WRITE
F	1402	FILE filename DOES NOT EXIST	SIL cannot perform the requested file function because the specified file does not exist.	Q5CHANGE Q5OPEN Q5PURGE Q5RETURN
W	1403	BAD "RSN" SPECIFIED	The user specified a request serial number that does not identify a pending I/O request. To obtain the number assigned to the request, specify the RSN= parameter on the Q5READ or Q5WRITE call.	Q5CHECK
F	1404	OVERLAPPING I/O REQUESTS TO SAME ADDRESS, FILE filename	The user attempted two or more I/O requests to the same address on the file without waiting for completion. Check the I/O for the completion before issuing multiple requests.	Q5CHECK
F	1405	FILE filename NOT CURRENTLY OPEN	The user has not opened the specified file for I/O. Issue a Q5OPEN call specifying the file.	Q5ENDPAR Q5GETN Q5GETP Q5PUTN Q5PUTP Q5READ Q5REWIND Q5SKIP Q5TRECov Q5WRITE
W	1406	FILE filename NOT CURRENTLY OPEN	The user attempted to close a file that is not open for I/O.	Q5CLOSE
W	1407	INCONSISTENT 'NAB VALUE', FILE IS CLOSED	The relative bit address, maintained by SIL, of the next byte to be written in the file, did not correspond to that maintained by the system at the block level. The file is closed. Consult an analyst.	Q5CLOSE
W	1408	FILE filename IS OPEN TO ANOTHER PROGRAM OF THIS USER	Because the specified file is open to another program executing under this user number, SIL can close the file for this program, but cannot destroy or give the file.	Q5CLOSE

Table B-2. System Utility Error Messages (Sheet 19 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1409	BUFFER SIZE OUT OF BOUNDS FOR FILE filename	The I/O buffer length must fit in 24 pages and be at least 1 block long. If the user is using the maximum buffer length of 24, the number of blocks per page, the buffer must be on a page boundary. Correct the buffer size.	Q5OPEN Q5CHECK Q5READ Q5WRITE
F	1411	LARGE-PAGE BUFFER FOR FILE filename IS GREATER THAN 128 *24 BLOCKS	The I/O buffer length must fit in 24 pages and be at least 1 block long. If the user is using the maximum buffer length of 24, the number of blocks per page, the buffer must be on a page boundary. Correct the buffer size.	Q5CHECK Q5OPEN Q5READ Q5WRITE
F	1412	ILLEGAL SKIP ON U-TYPE FILE filename	SIL cannot skip logical partitions (records, groups or files) on a U format file. It can skip physical blocks.	Q5SKIP
F	1413	SKIP FORWARD ILLEGAL. LAST OP. WAS WRITE ON FILE filename	The user cannot issue a Q5SKIP call to skip forward on a file when the last operation on the file was a write operation. The user can request a skip backward.	Q5SKIP
W	1414	BEGINNING OF INFORMATION ENCOUNTERED, FILE filename	SIL cannot skip further backward as it has reached the beginning of information for the file.	Q5SKIP
F	1415	SKIP OF GROUPS ILLEGAL FOR F-TYPE FILE, NAME=filename	The F file format does not allow skipping by groups because group delimiters do not exist in F format.	Q5SKIP
F	1416	EOF ENCOUNTERED ON FILE filename	SIL has read to the end of the file.	Q5CHECK Q5GETN Q5GETP Q5READ Q5SKIP Q5WRITE

Table B-2. System Utility Error Messages (Sheet 20 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
W	1417	STATUS OF LAST I/O NOT CHECKED FOR FILE filename	The user did not check the status returned by the last I/O operation. If the Q5READ or Q5WRITE call did not specify the WAIT parameter, a Q5CHECK call must check its status. If the Q5READ or Q5WRITE call did specify the WAIT parameter, it must also specify the STATUS parameter indicating the variable to which status is returned.	Q5READ Q5WRITE
F	1421	FILE filename NOT IMPLICITLY OPEN	The user cannot issue a Q5MAPIN or Q5MAPOUT call for the specified file because it has not been opened for implicit I/O.	Q5MAPIN Q5MAPOUT
F	1422	FILE filename FUNCTION ABORTED BY THE STATION OPERATOR	The station operator aborted the function being performed on the specified file. If needed, request an explanation from the station operator.	Q5CHECK Q5ENDPAR Q5READ Q5SKIP Q5WRITE
F	1423	VIRTUAL FILE filename CANNOT BE LESS THAN 2 PAGES	The user cannot create a controllee file shorter than two small pages or reduce an existing controllee file to less than two small pages.	Q5DEFINE Q5REDUCE Q5RQUEST
F	1424	FILE filename IS STILL OPEN	SIL cannot perform the file request because the file is open to another task. Try again later.	Q5PURGE Q5REDUCE Q5RETFIT Q5RETURN
F	1425	FILE filename DOES NOT EXIST OR IS NOT ATTACHED	SIL (or TASKATT) cannot perform the requested operation on an unattached permanent file or a file that does not exist. The user must attach, define, or request the file.	Q5CHANGE Q5GIVE Q5OPEN Q5PERMIT Q5REDUCE Q5ROUTE TASKATT
W	1427	FILE filename IS ALREADY OPEN	SIL cannot open the file because it is already open.	Q5GETFIL Q5OPEN
F	1428	IMPLICIT MODE REQUIRED WITH MODDROP ON FILE filename	The caller specified the MODDROP parameter without specifying the IMP parameter to open the file for implicit I/O. Specify the IMP parameter and try again.	Q5GETFIL Q5OPEN

Table B-2. System Utility Error Messages (Sheet 21 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1429	filename MUST BE A VIRT. CODE FILE TO CHANGE THE DROPFILE LEN.	SIL cannot change the drop file length when the file is a physical data file. Remove the DFLEN= parameter from the Q5CHANGE call.	Q5CHANGE
F	1430	FILE filename NOT OPEN FOR EXPLICIT I/O	Either the file is not open or it is open for implicit I/O.	Q5ENDPAR Q5GETN Q5GETP Q5PUTN Q5PUTP Q5READ Q5WRITE
F	1432	NO WSA= DEFINED FOR FILE filename	The user cannot perform explicit I/O by logical partitions without specifying a working storage area for the file.	Q5GETN Q5GETP Q5PUTN Q5PUTP
F	1433	WSA LOCATION NOT IN VALID RANGE FOR FILE filename	The memory address for the WSA is in the register file area or in the virtual system range.	Q5PUTN Q5PUTP Q5PUTB
F	1434	END OF INFORMATION ENCOUNTERED ON FILE filename	SIL either attempted to read past the end of the file or attempted to write past the maximum length of the file. If the file is a direct access file, ensure that the specified record number is correct.	Q5CHECK Q5ENDPAR Q5GETN Q5GETP Q5PUTN Q5PUTP Q5READ Q5SKIP Q5WRITE
F	1435	RECORD LENGTH OUTSIDE MIN/MAX RANGE FOR FILE filename	SIL transferred a record shorter or longer than the range of record lengths specified by the mnr and mxr fields in the FIT.	Q5GETN Q5GETP Q5PUTN Q5PUTP
W	1436	DATA QUANTITY EXCEEDS WSL FOR FILE filename	The length of the partition requested exceeds the working storage area length. SIL truncated the partition, discarding the excess data.	Q5GETN
F	1437	GET FOLLOWS OUTPUT OPERATION ON FILE filename	The user cannot issue a Q5GETN or Q5GETP call for the specified file because the last operation on the file was an output operation.	Q5GETN Q5GETP Q5READ
F	1438	CONTROL WORD PARITY ERROR ON FILE filename	SIL read a control word with a odd parity error. This could indicate that the file is not a W format file.	Q5GETN Q5GETP Q5SKIP

Table B-2. System Utility Error Messages (Sheet 22 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1439	CONTROL WORD FIELD ERROR ON FILE filename	While checking the control word linkage of the records within the group, SIL read a control word with a field error. Either the control word was written incorrectly or the file does not contain W format records.	Q5ENDPAR Q5PUTN Q5PUTP
W	1440	END OF RECORD ENCOUNTERED ON FILE filename	While reading partial records, SIL read the end of the current record.	Q5GETP
W	1441	END OF GROUP ENCOUNTERED ON FILE filename	SIL has read to the end of the group.	Q5GETN Q5GETP Q5SKIP
F	1442	WRITE NOT ALLOWED ON FILE filename; NO WRITE ACCESS	SIL cannot write on the specified file because it was not opened for write access.	Q5WRITE
F	1443	PUT NOT ALLOWED AFTER EOF ON FILE filename; R/U TYPE RECORDS	SIL cannot write data after the end of file in a U format file because the U format does support file delimiters.	Q5PUTN Q5PUTP
F	1444	PART=GROUP FOR RT=U or F ILLEGAL; FILE filename	SIL cannot read, write or skip groups on the specified F or U format file because the F and U formats do not have group delimiters.	Q5ENDPAR Q5GETN Q5GETP Q5PUTN Q5PUTP Q5SKIP
F	1445	PART=GROUP FOR RMK NOT 1F OR 1E ILLEGAL; FILE filename	The R record format does not support groups if the record mark character is not ASCII US (#1F) or RS (#1E).	Q5GETN Q5GETP
W	1448	MAX LENGTH OF FILE filename LESS THAN REQUESTED LENGTH	The user specified a file length on the Q5GETFIL call longer than the maximum allowed file length. Specify a smaller file length.	Q5GETFIL
F	1450	NO SPACE IN FIT FOR ENTRY FOR FILE filename	No more FITs can be generated for this user number until one or more existing FITs are discarded.	Q5GENFIT
W	1451	MXR EXCEEDED ON FIXED FILE filename EXCESS DATA IGNORED	The working storage area length (WSL) added to the length of the data already written to the record exceeds the fixed record length (MXR). SIL transfers MXR characters and discards the excess data.	Q5PUTP

Table B-2. System Utility Error Messages (Sheet 23 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
W	1452	MXR UNSATISFIED ON FIXED FILE filename DATA RECORDS PADDED	For Q5PUTN, the working storage area length (WSL) is less than the fixed record length (MXR). For Q5PUTP, the WSL added to the length of the data written to the record is less than MXR. SIL pads the record to MXR characters.	Q5ENDPAR Q5PUTN Q5PUTP
F	1453	CANNOT DEFINE CONNECTED FILE filename	The name of a file connected to a terminal was specified on a define request. A connected file is created with a REQUEST control statement or a Q5RREQUEST call.	Q5DEFINE
W	1454	CANNOT CHANGE GIVEN ATTRIBUTES FOR dt FILE filename	The user specified attribute changes that cannot be performed for the specified device type. The invalid parameters are ignored; the valid attribute changes are performed.	Q5CHANGE
F	1455	CANNOT CREATE dt FIT FOR FILE filename	The device type specified on the generate FIT request is not the same as the actual device type of the file specified on the request. Correct the file name or the device type.	Q5GENFIT
F	1456	GET PARTIAL RECORD FOR CONNECTED FILE filename NOT ALLOWED	The name of a file connected to a terminal was specified on a get partial record request. The only valid input request for a file connected to a terminal is a request for a full logical record (Q5GETN).	Q5GETP
F	1457	GIVE NOT ALLOWED FOR CONNECTED FILE filename	The name of a file connected to a terminal was specified on a give file request. The user cannot change the ownership of a file connected to a terminal.	Q5GIVE
F	1458	MAP OPERATION NOT ALLOWED ON CONNECTED FILE filename	The name of a file connected to a terminal was specified on a map in or map out request. A file connected to a terminal cannot be mapped in or mapped out.	Q5MAPIN Q5MAPOUT

Table B-2. System Utility Error Messages (Sheet 24 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1459	INELIGIBLE TO OPEN CONNECTED FILE filename AT THIS CHAIN LEVEL	The name of a file connected to a terminal was specified on an open request from a controllee not at level 2 of a controllee chain. Only a task initiated by an interactive execute line can open a connected file.	Q5GETFIL Q5OPEN
F	1460	CONNECTED FILE filename CANNOT BE OPENED IMPLICITLY	A file connected to a terminal cannot be opened for implicit I/O. Remove the IMP parameter from the Q5OPEN call.	Q5GETFIL Q5OPEN
F	1461	PUT OF PARTIAL RECORD FOR CONNECTED FILE filename NOT ALLOWED	The name of a file connected to a terminal was specified on a put partial record request. The only valid output request for a file connected to a terminal is a request for a full logical record (Q5PUTN).	Q5PUTP
F	1462	Q5READ CALL FOR CONNECTED FILE filename NOT ALLOWED	The name of a file connected to a terminal was specified on a read physical block request. The only valid input request for a file connected to a terminal is a request for a full logical record (Q5GETN).	Q5READ
F	1463	ROUTE NOT ALLOWED ON CONNECTED FILE filename	The name of a file connected to a terminal was specified on a route file request. The user cannot route a file connected to a terminal.	Q5ROUTE
F	1464	CONNECTED FILE NOT ALLOWED IN BATCH JOB	A batch job cannot create a file connected to a terminal. The request to create a connected file must be entered as an interactive execute line.	Q5RQUEST
F	1465	Q5WRITE CALL FOR CONNECTED FILE filename NOT ALLOWED	The name of a file connected to a terminal was specified on a write physical block request. The only valid output request for a connected file is a request for a full logical record (Q5PUTN).	Q5WRITE
F	1466	ILLEGAL MESSAGE DETECTED BY THE NAD/STATION	A NAD or peripheral station has received an illegal message on a mass storage explicit I/O request.	Q5CHECK Q5READ Q5WRITE

Table B-2. System Utility Error Messages (Sheet 25 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1467	FATAL DEVICE ERROR DETECTED BY THE NAD/STATION	A NAD or peripheral station has received a fatal device error on a mass storage explicit I/O request.	Q5CHECK Q5READ Q5WRITE
F	1468	INVALID COMPRESSED BLANK COUNT IN FILE filename	An error has occurred in expanding blank compression information, or in putting a #1B with an illegal blank compression count. A blank count number, reduced by the #30 bias, is less than zero and therefore invalid.	Q5GETN Q5GETP Q5PUTN Q5PUTP
F	1469	USER NOT VALIDATED FOR TAPE USAGE	The user has requested a tape; but is not validated for tape access.	Q5RQUEST
W	1470	RPB - NOT APPLICABLE FOR FILE filename	A records per block (RPB) value was specified for a tape file whose blocking type is not K (record count blocking). The RPB value is ignored.	Q5CHANGE Q5GENFIT
W	1471	MPRU - NOT APPLICABLE FOR FILE filename	An MPRU size value was specified for a tape file whose tape format is not V (variable). The MPRU value is ignored.	Q5CHANGE Q5GENFIT
W	1472	FILE filename PARAMETERS IGNORED - INCONSISTENT WITH DT=device type	The user specified parameters for a nontape file operation that are applicable to tape files only. The tape parameters are ignored.	Q5CHANGE Q5CHECK Q5GENFIT Q5OPEN Q5READ Q5RETURN Q5RQUEST Q5SKIP Q5WRITE
W	1473	LABEL PARAMETERS IGNORED FOR FILE filename	The user-specified label parameters for a file that is not an ANSI standard labeled file. The label parameters are ignored.	Q5CLOSE
W	1474	LABEL BUFFER TOO SHORT FOR FILE filename	The specified label array length is shorter than all labels to be copied. Labels are copied to fill the array.	Q5CLOSE Q5OPEN Q5REELSW
F	1475	ILLEGAL USER LABELS ENCOUNTERED FOR FILE filename	The user-specified label array contains invalid labels. The valid label formats are shown in appendix F of this manual.	Q5CLOSE Q5OPEN Q5REELSW

Table B-2. System Utility Error Messages (Sheet 26 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1476	IOER=code ENCOUNTERED ON FILE filename	The tape I/O error corresponding to the specified code has occurred for the file. The error code meanings are listed in table B-4. To clear the tape error status, call the Q5CLIOER routine.	Q5CHECK Q5CLOSE Q5GETFIL Q5OPEN Q5READ Q5RELSW Q5SKIP Q5WRITE
F	1479	TAPE FILE filename DOES NOT EXIST	The specified tape file has not been requested. A tape file can be requested with a REQUEST control statement or a Q5RQUEST call.	Q5GETFIL
F	1482	HDR1 LABEL NOT FOUND FOR TAPE FILE filename	An attempt to open an ANSI labeled tape file for read access could not find a HDR1 label on the tape. Either the file is an unlabeled file or it contains no data.	Q5OPEN
F	1483	ILLEGAL FSN FOR TAPE FILE filename	The specified file sequence number (FSN) is invalid. Valid numbers range from 1 through 9999.	Q5GETFIL Q5OPEN Q5RTPOS Q5TRECov
F	1484	ILLEGAL COMBINATION OF RT/BT FOR dt FILE filename	The specified combination of record type and blocking type is invalid for the specified device type. The valid combinations for tape files are shown in figure 9-2, Q5CHANGE Call Format.	Q5GETFIL Q5OPEN
F	1485	FILEID/SEQNO NOT FOUND FOR TAPE FILE filename	A HDR1 label containing the specified file identifier and/or file sequence number cannot be found in the multifile set.	Q5GETFIL Q5OPEN
F	1486	BUFFER SIZE IS .LT. MPRU FOR TAPE FILE filename	The specified buffer is smaller than the MPRU size for the tape file. Increase the buffer length.	Q5GETFIL Q5OPEN Q5READ
W	1487	ADO OPTION IGNORED FOR CODED TAPE FILE filename	The assembly/disassembly option (ADO) is invalid if the data conversion option (CONVERT) has been selected. Remove either ADO or CONVERT.	Q5GETFIL Q5OPEN
F	1488	BUFFER SIZE GREATER THAN 48 PAGES FOR TAPE FILE filename	The specified buffer is more than 48 pages long. Decrease the buffer size.	Q5READ

Table B-2. System Utility Error Messages (Sheet 27 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1489	EOT ENCOUNTERED ON TAPE FILE filename	The I/O request has encountered the end of the tape volume. The I/O operation is terminated. At this point, a Q5REELSW call could write additional volume labels. Another I/O request must be processed to continue the I/O operation on the next tape volume.	Q5READ Q5SKIP Q5WRITE
F	1490	CONVERSION DOES NOT MATCH LABEL FOR TAPE FILE filename	The specified character conversion mode (CM) does not match the character conversion mode used to write the tape labels. Correct the CM parameter or overwrite the tape labels.	Q5REQUEST
F	1491	NO VSN LIST SPECIFIED FOR filename	The tape file request did not specify the tape volumes for the file. Specify an array on the VSNA= parameter that contains one or more VSNs, one VSN per word.	Q5REQUEST
F	1492	MORE THAN 255 VSN'S SPECIFIED FOR filename	The VSN list contains more than 255 VSNs. Decrease the number of VSNs.	Q5REQUEST
F	1493	ILLEGAL CONVERSION MODE FOR filename	The only valid character conversion modes are ASCII (AS) and EBCDIC (EB). Correct the CM specification.	Q5REQUEST
F	1494	ATTEMPT TO READ PRU LONGER THAN MPRU FOR FILE filename	The user attempted to read a PRU longer than the MPRU size for the file. Either decrease the number or length of the LRUs read or increase the buffer length.	Q5CHECK Q5READ
F	1495	ILLEGAL DENSITY SPECIFIED FOR filename	The valid recording densities are 1600 cpi (PE) and 6250 cpi (GE). Correct the density specification.	Q5REQUEST
F	1496	MULTIFILE SET filename SHOULD BE ANSI - STD LABELED	The tape requested is unlabeled or nonstandard. A multifile set must use ANSI standard labels to delimit the files in the set.	Q5LABEL
F	1497	ILLEGAL VSN'S SPECIFIED FOR filename	One or more of the specified VSNs is invalid. A valid VSN is six characters long.	Q5REQUEST

Table B-2. System Utility Error Messages (Sheet 28 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1498	READ/SKIP FORWARD AFTER WRITE ON FILE filename	The user attempted to reposition the tape file forward after writing on the file. A tape file cannot be positioned beyond the end of the file data.	Q5SKIP
F	1499	ATTEMPT TO WRITE PRU LONGER THAN MPRU FOR FILE filename	The user attempted to write a PRU longer than the MPRU size for the file. Either increase the MPRU size or decrease the amount of data written.	Q5WRITE Q5CHECK
F	1500	SUM OF LRU'S GREATER THAN BUFFER LENGTH FOR FILE filename	The sum of the LRU lengths in the LRU description array is longer than the buffer length. Either decrease the number or length of the LRUs written or increase the buffer length.	Q5CHECK
F	1501	FILE filename IS IN USE BY ANOTHER JOB	SIL cannot attach the specified file because it is currently attached to another job in write, append, or modify mode.	Q5ATTACH
W	1502	FILE filename ALREADY EXISTS AS A LOCAL FILE	SIL cannot attach the specified permanent file because a local file with that name is currently assigned to the job. If the attach request was for all unattached files belonging to the user (Q5ATTACH,*), no filename is returned; the characters OR > are returned in the filename field.	Q5ATTACH
W	1503	POOL NAME poolname ALREADY ATTACHED	SIL cannot attach the specified pool because it is already attached to the job.	Q5PATACH Q5PREACC
W	1504	FILE filename ALREADY ATTACHED AS A PERMANENT FILE	SIL cannot attach the specified permanent file because it is already attached to the job.	Q5ATTACH

Table B-2. System Utility Error Messages (Sheet 29 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1505	FILE filename ALREADY EXISTS	SIL cannot create a file with or change a file's name to the specified file name because a file with that name already exists.	Q5CHANGE Q5DEFINE Q5REQUEST
W	1506	POOL NAME pname IS NOT ATTACHED	SIL cannot detach the specified pool because it is not attached to the job.	Q5PDTACH
F	1507	DUPLICATE POOL NAME pname	SIL cannot add the specified pool name to the pool list because it already exists in the pool list.	Q5PCREAT
F	1508	INVALID POOLNAME pname	The specified pool name does not conform to the pool naming conventions (one through eight characters, beginning with a letter).	Q5PATACH Q5PCREAT Q5PDESTR Q5PERMIT Q5PGRACC Q5PREACC Q5PUSERL
F	1509	INVALID USER NUMBER	The user specified the input queue manager user number on the Q5GIVE call. To give a file to the input queue manager, the user must specify the IQM and ACCT= parameters.	Q5GIVE
F	1510	UNABLE TO ATTACH ALL FILES	SIL could not attach one or more of the files belonging to the user number.	Q5ATTACH
F	1511	NO ROOM IN SYSTEM FOR ANY MORE POOLS	The user attempted to create a pool, but the system list of pools (Q5POOLS) is full. Until a PDESTROY is done on a pool, no new pools can be created.	Q5PCREAT
W	1512	POOL pname CANNOT BE DETACHED BECAUSE OF OPEN FILES	SIL cannot detach the specific pool because there are files open from the pool.	Q5PDTACH
F	1515	CANNOT CREATE TAPE FILE filename	The user attempted to create a tape file; tape files are not supported by the current VSOS version.	Q5REQUEST
F	1517	ILLEGAL USER NUMBER FOR USER1 FUNCTION ON FILE filename	The user number does not have User-1 privileges, but was attempting a User-1 function on the specified file.	Q5CLOSE Q5OPEN

Table B-2. System Utility Error Messages (Sheet 30 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1518	CANNOT PRIVILEGE OPEN ATTACHED FILE filename	A nonprivileged user attempted a privileged open.	Q5OPEN
F	1519	VIRTUAL ADDRESS OVERLAP ON FILE filename	Either the user specified an area of virtual space to be mapped in to the specified file that is already mapped in to another file or specified an area to be mapped out that is not mapped in to the drop file. Check that the array was properly specified on GROS parameter of LOAD statement.	Q5MAPIN Q5MAPOUT
F	1521	FILE filename NOT AT EOI IN APPEND MODE	The caller attempted to write on a file that is open for append access but is not positioned at the end of the file data. Reposition the file at its end and try again. (Read access permission is required to position the file.)	Q5PUTN Q5PUTP Q5WRITE
F	1522	UNABLE TO PROCESS FILE filename, TOO MANY ACTIVE FILES	SIL cannot create or open the specified file because 70 files (the operation system limit) are already active for this task.	Q5DEFINE Q5OPEN Q5REQUEST
F	1523	NEW FILENAME IS INVALID	The new filename specified in the Q5CHANGE call is invalid.	Q5CHANGE
F	1524	CANNOT CREATE FILE WITH HIGHER SECURITY LEVEL	A call to create a file specified a security level greater than the security level of the calling task. Request a lower security level for the new file.	Q5DEFINE Q5REQUEST
F	1525	SECURITY LEVEL OF FILE filename TOO HIGH	If the call attempted to open the file, the security level of the file is higher than the maximum security level allowed the caller. If the call attempted to give the file to a pool or to another user, the security level of the file is higher than the maximum security level allowed the pool boss or the other user.	Q5GIVE Q5OPEN
F	1526	USER DIRECTORY OR POOL WAS NOT FOUND FOR FILE filename	The owner of filename was not found in the user directory or poollist.	Q5OPEN Q5GETFIL
F	1529	FILE filename NOT OPENED; NO ROOM IN USER TABLE	The system cannot enter the specified file in the user activity table. Consult a systems analyst.	Q5OPEN

Table B-2. System Utility Error Messages (Sheet 31 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1531	UNABLE TO DESTROY POOL pname	The user cannot destroy the specified pool for one of the following reasons: <ul style="list-style-type: none"> • The user is not the pool boss. • Another job has the pool attached. • Files still exist that belong to the pool. 	Q5PDESTR
F	1532	SELECT=P IS REQUIRED WHEN DEVICE=0 IS SPECIFIED	The user has selected DEVICE=0 but has not selected SELECT=P	DUMPF
F	1533	DEV=0 AND VSN= PARAMETERS ARE MUTUALLY EXCLUSIVE	The user has selected DEVICE=0 and has specified the VSN parameter. These parameters should not both be specified.	DUMPF
F	1536	MASS STORAGE FILE INDEX TABLE ENTRY NOT FOUND	File was not found in system file index table. Contact a systems analyst.	Q5MAPIN
F	1537	NOT ENOUGH ROOM IN DROFFILE	The user can map no more virtual space into the drop file.	Q5MAPIN
F	1538	CANNOT MAPIN FILE filename AT VIRTUAL PAGE ZERO	The user attempted to map virtual space beginning at the first page (virtual page zero). Virtual page zero is reserved for the register file. The user can begin to map space at the second page (virtual page one).	Q5MAPIN
F	1541	FILE INDEX COPY FOR FILE filename IS OUT OF BOUNDS	The user specified an array containing the File Index entry that is outside the virtual address space the user is permitted to access.	Q5OPEN
F	1542	CANNOT ATTACH POOL pname - FILE INDEX IS FULL	Other users must destroy some of their files or log off to free space in the file index. Reenter the PATTACH command to attach POOL.	Q5PATTACH

Table B-2. System Utility Error Messages (Sheet 32 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1543	FILE CURRENTLY IN USE BY A PRIVILEGED TASK	You have specified WAIT=N or an access permission of write, modify, or append on an ATTACH or Q5OPEN which cannot access the file because a system utility has the file open. Try again later or change the access permission.	Q5ATTACH Q5OPEN
F	1544	CANNOT ATTACH pname, ALREADY ATTACHED TO 4 POOLS	The job has four pools attached and so cannot attach another pool. To attach the specified pool, detach one of the attached pools.	Q5PATACH
F	1545	CANNOT ATTACH POOL pname - USER HAS NO ACCESS	The pool boss for the specified pool has not granted pool access to this user number. Request pool access from the pool boss.	Q5PATACH
W	1546	USER IS NOT THE POOL BOSS FOR POOL FILE filename	The user cannot perform the requested file function because he is not the pool boss of the pool that owns the file. Check that the correct file name is specified on the call.	Q5PERMIT Q5PURGE
F	1547	USER IS NOT THE POOL BOSS FOR POOL pname	The user attempted to perform a pool management function for a pool for which he is not the pool boss. Check that the correct pool name is specified on the call.	Q5GIVE Q5PDESTR Q5PGRACC Q5PREACC
F	1548	NO READ ACCESS SPECIFIED FOR FILE filename	The user cannot map in a file that is not opened for read access.	Q5MAPIN
F	1549	DATA EXCEEDS USER SPECIFIED LENGTH OF BUFFER	The buffer specified on the call is not large enough to hold all the pool list entries. Increase the size of the buffer.	Q5POOLS
F	1551	MASS STORAGE ADDR+LENGTH EXCEEDS LENGTH OF FILE filename	The virtual region length starting at the specified mass storage block exceeds the length to which the file can extend.	Q5MAPIN Q5MAPOUT
F	1553	A VIRTUAL ADDRESS OF FILE filename NOT ON PAGE BOUNDARY	The specified virtual address is not on a page boundary. Correct the virtual address so that it is a multiple of the page size.	Q5MAPIN Q5MAPOUT

Table B-2. System Utility Error Messages (Sheet 33 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1560	FUNCTION FAILED FOR FILE filename; BOUND IMPLICIT MAP FULL	The user cannot map another virtual address range into the specified file without mapping out a mapped in range or combining two mapped in ranges.	Q5MAPIN Q5MAPOUT Q5OPEN
F	1561	CANNOT PERFORM FUNCTION ON FILE filename; PAGES STILL LOCKED IN	SIL cannot map out the virtual address range because it is currently performing implicit I/O from that region. If the user is nonprivileged, this message could indicate a system error. Consult a systems analyst.	Q5MAPOUT
F	1562	SPACE UNDEFINED AT MAPOUT FOR FILE filename	SIL cannot map out the virtual address range because that range is not mapped to the specified file.	Q5MAPOUT
F	1565	INCORRECT LENGTH OF VIRTUAL REGION FOR FILE filename	The user specified a longer virtual address range to be mapped out than the range originally mapped in.	Q5MAPOUT
F	1566	DROP FILE MAP FULL	SIL cannot map out another virtual address range to the drop file.	Q5CLOSE Q5GETFIL Q5MAPOUT
F	1570	ERROR IN MODIFYING THE PFI ENTRY FOR FILE filename	SIL encountered an error while attempting to modify the PFI entry for the file filename. Try again; if the error reoccurs, notify a systems analyst.	Q5OPEN Q5GIVE

Table B-2. System Utility Error Messages (Sheet 34 of 80)

Severity	Error Code	Message	Significance	Issuing Routine																
F	1580	MULTIPLE STATION (SERR) ERROR CODES; VALUE=value	<p>The NAD detected more than one error condition. SIL combined the codes for the error conditions, using inclusive OR operations to form the value in the message. Each bit set indicates an error condition. The hexadecimal values are as follows:</p> <table data-bbox="889 716 1328 1199"> <thead> <tr> <th><u>Value</u></th> <th><u>Condition</u></th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Device not ready.</td> </tr> <tr> <td>2</td> <td>Transmission parity error.</td> </tr> <tr> <td>10</td> <td>End of file encountered.</td> </tr> <tr> <td>40</td> <td>Disk channel failed.</td> </tr> <tr> <td>200</td> <td>Mass storage positioning error.</td> </tr> <tr> <td>400</td> <td>Operator aborted function.</td> </tr> <tr> <td>800</td> <td>File extension error.</td> </tr> </tbody> </table>	<u>Value</u>	<u>Condition</u>	1	Device not ready.	2	Transmission parity error.	10	End of file encountered.	40	Disk channel failed.	200	Mass storage positioning error.	400	Operator aborted function.	800	File extension error.	Q5REWIND Q5SKIP
<u>Value</u>	<u>Condition</u>																			
1	Device not ready.																			
2	Transmission parity error.																			
10	End of file encountered.																			
40	Disk channel failed.																			
200	Mass storage positioning error.																			
400	Operator aborted function.																			
800	File extension error.																			

Table B-2. System Utility Error Messages (Sheet 35 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1586	CANNOT CHANGE REQUESTED ATTRIBUTE FOR FILE filename	File filename is a tape file and the requested attribute change is valid only for mass storage files.	Q5CHANGE
F	1589	DEVICE NOT READY FOR FILE filename	The device on which the file resides is not ready to transfer data. Request that the operator ready the device.	Q5CHECK Q5READ Q5WRITE
F	1596	ERROR IN POSITIONING MASS STORAGE DEVICE FOR FILE filename	Hardware error. Consult a systems analyst.	Q5CHECK Q5READ Q5REWIND Q5SKIP Q5WRITE
F	1598	CANNOT PERFORM FUNCTION ON FILE filename - I/O CHANNEL FAILED	SIL could not perform the requested function because the I/O channel failed. Consult a systems analyst.	Q5CHECK Q5READ Q5WRITE
F	1604	FATAL I/O ERROR ON FILE filename PACK packname	Hardware error. Consult a systems analyst.	Q5CHECK Q5READ Q5WRITE
F	1605	SMALL AND LARGE PAGES EXIST IN THE BUFFER.	The buffer specified for I/O must be either small or large pages but not both.	Q5CHECK Q5READ Q5WRITE
F	1606	BUFFER ALREADY IN USE	The buffer is already in use for explicit I/O. Call Q5CHECK to determine if the I/O using this buffer is completed.	Q5READ Q5WRITE
F	1615	ILLEGAL LABEL FOR TAPE FILE filename	In an attempt to verify the HDR1 label, labels were found in the wrong order, there was an unknown label, or there were illegal characters in the HDR1 label.	Q5LABEL

Table B-2. System Utility Error Messages (Sheet 36 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1616	DATA MAY BE LOST DUE TO FILE TRUNCATION	The file is only partially available.	Q5GETFIL Q5OPEN
F	1617	READ ONLY ACCESS REQUIRED FOR PARTIAL FILE ATTACH	The job has requested something other than read-only access on a file for which not all segments are available.	AUDIT Q5ATTACH DUMPF Q5OPEN ATTACH
W	1618	SOME FILES ARE INCOMPLETE DUE TO DEVICE(S) DOWN	Rerun when devices are up.	Q5PATACH Q5ATTACH
F	1619	SOME FILE PARTS UNAVAILABLE FOR FILE	Not all segments of file are available. Try again when necessary devices are up.	AUDIT PATTACH Q5ATTACH Q5OPEN ATTACH DUMPF
F	1620	ATTEMPT TO IMPLICITLY OPEN FILE filename WITH WRITE ONLY ACCESS	SIL cannot open a file for implicit I/O to which the user does not have read access.	Q5OPEN
F	1624	LABEL CALL ILLEGAL FOR FILE filename	An HDR1 label was specified for a file that is not an ANSI labeled file. Either change the label type of the file or access it as an unlabeled or nonstandard labeled file.	Q5LABEL
F	1625	MULTIFILE SET filename DOES NOT EXIST FOR FILE filename	The specified multifile set has not been requested. A REQUEST control statement or Q5RQUEST call to request the multifile set must precede any label specification for a file in the set.	Q5LABEL
F	1627	DUPLICATE FSN SPECIFIED	The same FSN was specified a second time. Check previous LABEL/Q5LABEL calls.	Q5LABEL
W	1628	NOT ALL TAPES TABLE ENTRIES RETURNED FOR TAPE FILE filename	The array to which the tapes table entry was to be copied is less than 12 words. The entry is copied up to the length of the array.	Q5OPEN
W	1629	NOT ALL VSN ENTRIES RETURNED FOR FILE filename	The VSN list array is not long enough for all VSNs in the VSN list. VSNs were copied up to the length of the array.	Q5OPEN

Table B-2. System Utility Error Messages (Sheet 37 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
W	1630	SPECIFIED ACCESS INVALID FOR TAPE FILE filename	Append, modify, and execute access modes are invalid for a tape file. Only read and write access modes are valid.	Q5PERMIT
F	1633	SUM OF LRU'S .GT. BUFFER LENGTH FOR TAPE FILE filename	The sum of LRU sizes specified in the LRU array exceeds the buffer length for this file. Increase the buffer length, or decrease the sum of the LRU sizes.	Q5WRITE
F	1635	'FA' MISMATCH ON FILE filename	The file access field in the tape label does not match the FA parameter specified. Correct the parameter.	Q5OPEN
F	1637	REQUESTING TAPE FILE filename RESULTS IN OVERCOMMITMENT	Correct the NT parameter in resource.	Q5REQUEST
F	1639	ILLEGAL REEL NUMBER FOR TAPE FILE filename	The system tapes table entry contains an invalid reel number.	Q5RTPOS Q5TRECov
F	1640	ILLEGAL SECTION NUMBER FOR TAPE FILE filename	The system tapes table entry contains an invalid section number.	Q5RTPOS Q5TRECov
F	1644	POOL NAME pname IS NOT DEFINED OR CALLER NOT POOL MEMBER	The user cannot give a file to the pool either because the pool is not attached or because the pool boss has not granted the user access to the pool.	Q5GIVE
F	1650	BUFFER NOT ON 512-WORD PAGE BOUNDARY FOR FILE filename	The I/O buffer for the specified file is not on a page boundary. The user must specify the LOAD utility parameter to load the buffer on a page boundary.	Q5GENFIT Q5OPEN Q5READ Q5SETFIT Q5WRITE
F	1653	BUFFER FOR FILE filename IN UNASSIGNED VIRTUAL SPACE	A possible extension to a file mapped in for implicit I/O could overlap an explicit I/O buffer. Group the common blocks containing the explicit I/O buffer at a higher address so it cannot overlap a file extension.	Q5CHECK Q5READ Q5WRITE
F	1679	NEW LENGTH FOR FILE filename GREATER THAN EXISTING FILE	The user specified the reduced file length to be greater than the existing file length.	Q5PERMIT Q5REDUCE

Table B-2. System Utility Error Messages (Sheet 38 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1680	FILE filename ALREADY EXISTS AT DESTINATION	The user cannot give the specified file as requested because the destined owner (another user, a pool, or the public file list) already has a file with that name.	Q5GIVE
F	1682	UNDEFINED USER NUMBER usernum	The user specified a nonexisting or invalid user number.	Q5ATTACH Q5CHANGE Q5GIVE Q5PERMIT
F	1683	OUTPUT FILE filename IMPROPERLY NAMED	Print file names must be in the format Pnnxxxxx where nn is two digits indicating the position of the file within a family of print files and xxxxx is the family name.	Q5GIVE
F	1685	FUNCTION FAILED ON FILE filename USER NOT PRIVILEGED	The DUMP parameter on Q5CLOSE can be specified only by a privileged user. Remove the parameter.	Q5CLOSE
F	1686	FILE filename IS A CONTROLLEE OR DROPFILE	SIL cannot transfer ownership of a controllee file or a drop file.	Q5GIVE
F	1687	DISK IS LOGICALLY OFF FOR FILE filename	The disk on which the file resides is not currently available to the system. Ask the operator to logically turn on the disk.	Q5DEFINE Q5GIVE Q5OPEN Q5PURGE Q5RETURN Q5REQUEST Q5ATTACH Q5CHANGE Q5ROUTE
F	1688	FUNCTION ON FILE filename WOULD EXCEED FILE SPACE LIMIT	SIL did not create a local file or give a file as requested by the user because the file space of the user would be exceeded.	Q5DEFINE Q5GIVE Q5REQUEST
F	1689	VRI= NOT VALID FOR NON-CODE FILE filename	The user cannot specify a variable rate index for a data file.	Q5GIVE
F	1690	FILE filename IS IN USE	SIL cannot perform the file request because the file is open. Close the file and try again.	Q5CHANGE Q5GIVE Q5PURGE Q5RETURN Q5ROUTE

Table B-2. System Utility Error Messages (Sheet 39 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1691	USER filename IS NOT OWNER OF FILE	The user cannot perform the requested file function because he is not the owner of the file.	Q5CHANGE Q5GIVE Q5PERMIT Q5PURGE Q5REDUCE Q5ROUTE
F	1692	INVALID ACCOUNT IDENTIFIER	The account identifier specified is not a valid account for the user. Correct the account identifier.	Q5GETUID Q5LFIPOL Q5LFIPRI Q5LFIPUB Q5CHANGE
F	1693	ILLEGAL MASTER PROJECT NUMBER	The master project number specified does not consist of one to three alphanumeric characters. Correct the MPN.	Q5CHANGE
F	1694	FILE filename is ATTACHED BY ANOTHER JOB	The user cannot purge the file until all jobs which have the file attached have completed.	Q5PURGE
F	1700	NO DISPOSITION SET FOR FILE filename	To route a file, the user must specify a disposition code for the file using the DC= parameter.	Q5ROUTE
F	1701	ILLEGAL DISPOSITION CODE xx	SIL does not recognize the specified disposition code.	Q5GETFIL Q5ROUTE
F	1702	ILLEGAL SITE IDENTIFIER site, OR SITE NOT LOGGED IN	SIL either does not recognize the site identifier or the specified site is not logged in to the system.	Q5ROUTE
F	1704	NO ROOM FOR USER TABLE, FILEI OR FST ENTRIES	The user activity table or the FILEI or FST system tables have no more available space at this time. This is usually due to a very heavy system load. Try again at a time when the system load is lighter.	Q5DEFINE Q5CHANGE Q5PERMIT
F	1707	LOGICAL FILE ADDRESS OVERLAP	Mapping addresses overlap; change the addresses.	Q5MAPIN Q5MAPOUT
F	1708	ERROR IN EXTENDING FILE	A system error has occurred during file extension. Try using another file on a different device set.	Q5READ Q5WRITE Q5CHECK Q5MAPIN Q5MAPOUT

Table B-2. System Utility Error Messages (Sheet 40 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1709	FILE LENGTH EXCEEDS TARGET USER'S OR POOL'S ALLOWED MAXIMUM	The file to be given to user or pool exceeds maximum file size for that recipient.	Q5GIVE GIVE
F	1710	FILE LENGTH EXCEEDS USER'S ALLOWED MAXIMUM	The user attempted to create or extend a file larger than the user's validated file size.	DEFINE Q5REQUEST Q5GETFIL Q5WRITE REQUEST Q5DEFINE
F	1711	NO MASS STORAGE SPACE FOR FILE filename	The system has no mass storage space available for creating or extending the specified file.	Q5DEFINE Q5REQUEST Q5GETFIL Q5WRITE Q5CHECK Q5MAPIN
F	1712	OPERATOR-INITIATED ERROR FOR FILE filename	The operator entered a command preventing creation of the specified file.	Q5DEFINE Q5REQUEST Q5GETFIL
F	1713	STANDBY JOB CANNOT REQUEST A TAPE FOR FILE filename	The job, running at too low a priority, requested a tape file. Increase the priority of the job and rerun.	Q5REQUEST
F	1716	CANNOT FIND DISK PACK pack FOR FILE filename	SIL cannot find an on-line disk whose name matches the pack name specified on the Q5DEFINE or Q5REQUEST call.	Q5DEFINE Q5GETFIL Q5REQUEST
F	1718	ATTEMPT TO EXCEED MAXIMUM ALLOWABLE FILE SIZE FOR FILE filename	The user specified a file length on the Q5DEFINE or Q5REQUEST call that exceeds the maximum length allowed by the installation.	Q5DEFINE Q5REQUEST
F	1719	WRITE BEYOND EOI IN MODIFY MODE FOR FILE filename	The user specified a write operation that would extend past the existing end of the file for a file opened for modify access. In modify mode, the file cannot be lengthened.	Q5ENDPAR Q5PUTN Q5PUTP Q5WRITE
F	1720	CANNOT LOCATE THE USER OR POOL, owner, FOR FILE filename	A privileged user specified an array on the Q5DEFINE call containing a file index entry, but the entry contains an unknown user number or pool name.	Q5DEFINE

Table B-2. System Utility Error Messages (Sheet 41 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1721	TAPE FILE filename CANNOT BE OPENED IMPLICITLY	The IMP parameter cannot be specified for a tape file. Eliminate the parameter.	Q5OPEN
F	1722	WITH FILE filename NUMBER OF FILES EXCEEDS LIMIT FOR USER	SIL cannot create or give the specified file because the number of files belonging to the user number would exceed the limit set for the user number.	Q5DEFINE Q5GIVE Q5RQUEST
F	1723	RHF filename CAN NOT BE ROUTED	The Q5ROUTE call specified a file with the RHF communication type. To route an output file from a job submitted via RHF, use the MFQUEUE control statement.	Q5ROUTE
F	1724	ILLEGAL OPERATION ON FILE filename	The requested operation (via SIL) cannot be performed on the file residing on this device. For example PURGE cannot be performed on a tape file. Eliminate the operation or change the device type for the file.	Q5CLIOER Q5GIVE Q5REDUCE Q5ROUTE Q5REELSW QPURGE
F	1725	FIT ALREADY EXISTS FOR FILE filename	The specified file name is already associated a FIT so the user cannot generate another FIT for that file.	Q5GENFIT Q5RQUEST
F	1726	ACCESS VIOLATION ON FILE filename	The request failed because the caller does not have the required access permission. If possible, add the access permission to the access permission set for the file and try again.	Q5ATTACH Q5ENDPAR Q5GETN Q5GETP Q5INIT Q5INITCH Q5OPEN Q5PUTN Q5PUTP Q5READ Q5SKIP Q5WRITE
F	1727	ILLEGAL ACCESS PERMISSIONS:	The caller specified an invalid access permission identifier. Correct the access permission set specification and try again.	Q5OPEN Q5PERMIT

Table B-2. System Utility Error Messages (Sheet 42 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1728	ACS NOT GIVEN - ACCESS DIRECTORY IS FULL FOR FILE filename	The user cannot specify another set of access permissions because the file access directory entry for the file is full. If possible, delete a set of access permissions and try again.	Q5PERMIT
F	1730	ACCESS CONFLICT WITH ANOTHER JOB ON FILE filename	A privileged user's job attempted either to access a file in read mode with the file already open in write, append, or modify mode, or to access a file in write, append, or modify mode with the file already open. Wait until the file is available and try again.	Q5ATTACH Q5GETFIL Q5INIT Q5INITCH Q5OPEN
F	1731	USER PARAM OR USERNO INVALID FOR FILE filename	The caller specified the USER parameter on a call that specifies a local file. Remove the USER parameter or correct the file name and try again.	Q5PERMIT
F	1732	CANNOT PRIVILEGE OPEN LOCAL FILE filename	Local files cannot be privileged opened. Use a nonprivileged open instead.	Q5OPEN
F	1733	CANNOT LOCATE VSN FOR TAPE FILE filename	One or more volumes described in the VSN descriptor list cannot be found in the system tables. Contact the site analyst.	Q5OPEN
F	1735	FILE filename IS INACCESSIBLE TO THIS SYSTEM RELEASE	The user attempted to access a file which was created on a system that is incompatible with the current system. Consult a systems analyst. If possible, recreate the file on the current system. (Seen on systems being converted to system 2.2 release.)	Q5ATTACH Q5GETFIL Q5INIT Q5INITCH Q5OPEN Q5PURGE
F	1736	CONTROLLEE REQUIRES USER DYN/SHARED LIB	This controllee requires the user dynamic or system shared library to be active before it can execute.	Q5INIT Q5INITCH
F	1737	CONTROLLEE MUST BE RELOADED	This controllee was loaded on a system prior to VSOS 2.2 and must be reloaded.	Q5INIT Q5INITCH
W	1738	CONTROLLEE USING WRONG LIBRARIES	The libraries the controller was loaded with are not available and different libraries have been linked. Make the original libraries available.	Q5INIT Q5INITCH

Table B-2. System Utility Error Messages (Sheet 43 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1761	FA=A AND USER NUMBER MISMATCH ON FILE filename	While opening a labeled (ANSI standard) tape file, the HDRI label contains file accessibility character of A and the first six characters of the ownerid does not match the user number under which the open was attempted.	Q5OPEN
F	1762	VOLUME NOT AVAILABLE FOR FILE filename	For tape file (filename) there is a FILEI entry but no tape unit has been assigned to this file.	Q5OPEN Q5REELSW Q5REQUEST
F	1763	ATTEMPTED WRITE ON UNEXPIRED TAPE FILE filename	The unexpired tape must be blank labeled before it can be written on.	Q5OPEN Q5REELSW Q5WRITE
F	1764	TRIED TO WRITE ZERO LENGTH RECORD OF V-FMT TAPE FILE filename	An LRU containing zero data cannot be written on V format tape.	Q5CHECK Q5WRITE
F	1765	INVALID OPERATION FOR THIS USER	Only privileged users can BLANK label a tape; this message indicates that a nonprivileged user tried to BLANK label a tape.	Q5REQUEST
W	1766	DENSITY MISMATCH ON TAPE FILE filename	Tape file filename was requested with a density other than that read from the VOL1 label for the VSN assigned. Request was completed normally. The density on the tape was used.	Q5REQUEST
F	1767	READ-ONLY ACCESS WHILE BLANK LABELING TAPE	User issued a direct Q5REQUEST call to BLANK label a tape specifying an access of read-only.	Q5REQUEST
F	1768	"VA" MISMATCH ON TAPE VOLUME	The volume access field in the tape label does not match the VA parameter specified. Correct the parameter.	Q5REQUEST
F	1769	INTERACTIVE TAPE ACCESS NOT ALLOWED	The installation parameter to allow interactive tape usage is set to zero at this site.	Q5REQUEST
F	1770	NON STANDARD LABEL USAGE NOT ALLOWED	The installation parameter to allow nonstandard labels is set to zero at this site.	Q5REQUEST

Table B-2. System Utility Error Messages (Sheet 44 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
W	1771	"RU" NOT ALLOWED ON TAPE FILE filename	The installation parameter to allow read unconditionally is set to zero at this site. Otherwise, the processing is completed normally.	Q5RQUEST
F	1772	ILLEGAL LABELS ENCOUNTERED WHILE BLANK LABELING TAPE	The labels (VOL1 and HDR1) passed to blank label a tape did not conform to ANSI standards.	Q5RQUEST
F	1773	TRIED TO WRITE TWO SUCCESSIVE TAPE MARKS ON FILE filename	On V-formatted tape file filename, user attempted to write two successive tape marks. The second tape mark was not written to the tape.	Q5RQUEST
F	1800	RECORD TYPE MUST BE FIXED FOR DIRECT FILE filename	The user specified direct access file organization (SFO=D) and a record format other than F format. Change either the record format or the file organization.	Q5CHANGE Q5DEFINE Q5GENFIT Q5GETFIL Q5OPEN Q5RQUEST
F	1801	DEVICE TYPE MUST BE MASS STORAGE FOR DIRECT FILE filename	When creating a file, the user specified direct access file organization (SFO=D) and a device type other than mass storage. Correct the device type or the file organization specification.	Q5GETFIL Q5RQUEST
W	1802	ILLEGAL VALUE FOR OSTAT FIELD	The calling task attempted to change the OSTAT field of the field of the FILEI to an illegal value.	Q5CHANGE
W	1803	UNABLE TO CHANGE OSTAT FIELD IN FILEI	The system was unable to change the OSTAT field in the FILEI.	Q5CHANGE
W	1804	PARAMETER(S) IGNORED FOR LOCAL filename	User specified SIO parameters when defining a local file. These parameters have been ignored.	Q5DEFINE
W	1805	I/O BUFFERS OVERLAP	The address of the lower I/O buffer plus its length is greater than the higher buffer address.	Q5GETFIL Q5GENFIL Q5SETFIT Q5OPEN
F	1810	NEGATIVE OR ZERO RECORD NUMBER FOR DIRECT FILE filename	The specified record number is not greater than zero. Specify an integer greater than zero as the record number.	Q5GETN Q5GETP Q5PUTN Q5PUTP

Table B-2. System Utility Error Messages (Sheet 45 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	1812	OUTPUT FILE lfn NOT GIVEN - NO JDNS AVAILABLE	The system has run out of available JDNS. Wait for a few seconds and try again. If the problem persists, contact a site analyst.	Q5GIVE
F	1813	INVALID LENGTH SPECIFIED ON Q5MAPIN CALL	The user specified a value for the LEN= parameter that was zero, greater than the length in the map, or was not modulo page size.	Q5MAPIN
F	1814	SPECIFIED JDN IS NOT CALLER-S JDN	The specified JDN must match the caller's JDN. Correct the problem and recompile.	Q5GIVE
F	1815	CALLING TASK IS NOT PRIVILEGED	The task is not allowed to use a privileged parameter.	Q5GIVE
F	1816	RECIPIENT IS NOT AN OUTPUT PROCESSOR	The task specified FIJDN= but the user receiving the file is not the user number for an output processor.	Q5GIVE
F	6001	TASKNAME GREATER THAN EIGHT CHARACTERS	The first word of the control statement is longer than eight characters. The first word names the task to be executed.	BATCHPRO
F	6002	NO TASK NAME SPECIFIED	The control statement began with a period or right parenthesis.	BATCHPRO
F	6003	EOF ON CONTROL CARD FILE BEFORE END OF COMMAND	The end-of-file indicator was encountered before a period or right parenthesis terminating the control statement. Add a period or right parenthesis to the control statement.	BATCHPRO
F	6004	INVALID CHARACTER SPECIFIED	The control statement contains a character that is not in the ASCII 64-character subset. Refer to appendix A of this manual for the characters in the subset.	BATCHPRO
F	6005	COMMAND OVER 4096 CHARACTERS LONG	The length of the control statement is greater than 4096 characters. Check that a period terminates the statement.	BATCHPRO

Table B-2. System Utility Error Messages (Sheet 46 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	6006	PROCEDURE pname NOT IN FILE filename	The specified file does not contain the specified procedure. Check the PROC statement in the file for the correct procedure name.	BATCHPRO
F	6007	KEYWORD SPECIFIED ON BEGIN STATEMENT NOT IN PROC STATEMENT	A BEGIN statement parameter uses a keyword that does not appear on the PROC statement. Correct the keyword to match a PROC statement formal parameter.	BATCHPRO
F	6008	BEGIN STATEMENT - TOO MANY PARAMETERS SPECIFIED	More substitution values are specified on the BEGIN statement than formal parameters on the PROC statement. Remove the excess parameters or commas.	BATCHPRO
F	6009	PROC STATEMENT - TOO MANY PARAMETERS SPECIFIED	More than sixteen formal parameters are specified on the PROC statement. Remove the excess parameters.	BATCHPRO
F	6010	BEGIN STATEMENT - POSITIONAL KEYWORD AFTER NONPOSITIONAL	A substitution value specified without a keyword appears after one or more parameters that specify keywords. Either specify the keyword with the substitution value or remove the keywords from the preceding parameters.	BATCHPRO
F	6011	MISSING PROC STATEMENT IN PROCEDURE FILE	The first statement of the procedure file is not a PROC statement. Add a PROC statement to the beginning of the file.	BATCHPRO
F	6012	PROC STATEMENT - MISSING PNAME	The PROC statement does not specify a procedure name. Insert a procedure name after the PROC verb.	BATCHPRO
F	6013	BEGIN STATEMENT - INVALID FILE NAME SPECIFIED	The file name specified on the BEGIN statement does not follow the rules for file names (one to eight letters and digits, beginning with a letter). Correct the file name.	BATCHPRO
F	6014	REQUESTED PROCEDURE FILE ALREADY IN USE	The procedure file is being executed by another job. Try again later.	BATCHPRO

Table B-2. System Utility Error Messages (Sheet 47 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	6015	PROC STATEMENT - END OF FILE ENCOUNTERED WHILE PROCESSING	BATCHPRO read the end of the file while reading the PROC statement of the procedure.	BATCHPRO
F	6016	BEGIN STATEMENT - KEYWORD OVER EIGHT BYTES LONG	A keyword specified on the BEGIN statement is longer than eight characters. Correct the keyword.	BATCHPRO
F	6017	PROC STATEMENT - KEYWORD OVER EIGHT BYTES LONG	A formal parameter specified on the PROC statement is longer than eight characters. Correct the formal parameter.	BATCHPRO
F	6018	BEGIN STATEMENT - KEYWORD VALUE OVER EIGHT BYTES LONG	A substitution value specified on the BEGIN statement is longer than eight characters. Correct the value.	BATCHPRO
F	6019	ATTEMPT TO ACTIVATE MORE THAN MAXIMUM NUMBER OF PROCEDURES	A procedure file eight nested levels from the batch input file contains a BEGIN statement. Change the eighth level procedure so that it does not call another procedure.	BATCHPRO
F	6020	BEGIN STATEMENT - PARAMETER SPECIFIED MORE THAN ONCE	The BEGIN statement specifies more than one substitution value for a PROC statement formal parameter. Remove the redundant parameter.	BATCHPRO
F	6021	PROC STATEMENT - PARAMETER SPECIFIED MORE THAN ONCE	The PROC statement specifies the same formal parameter more than once. Remove the redundant parameter.	BATCHPRO
F	6022	INTERNAL ERROR - PROCEURE FILE LEVEL ERROR	Error in a system routine. Notify a systems analyst.	BATCHPRO
F	6023	INTERNAL ERROR - NO MESSAGE FOR STATUS CODE status	Error in a system routine. Notify a systems analyst.	BATCHPRO
F	6024	BEGIN STATEMENT - PROCEDURE NAME TOO LONG	The procedure name specified on the BEGIN statement is longer than eight characters. Correct the procedure name.	BATCHPRO
F	6025	PROC STATEMENT - PROCEDURE NAME TOO LONG	The procedure name specified on the PROC statement is longer than eight characters. Correct the procedure name.	BATCHPRO

Table B-2. System Utility Error Messages (Sheet 48 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	6026	BEGIN STATEMENT - ILLEGAL SYNTAX	The syntax of the BEGIN statement is incorrect. Refer to the format shown in the BEGIN statement description in chapter 4 of this manual.	BATCHPRO
F	6027	PROC STATEMENT - ILLEGAL SYNTAX	The syntax of the PROC statement is incorrect. Refer to the format shown in the PROC statement description in chapter 4.	BATCHPRO
F	6028	FILE NAME filename CANNOT BE USED FOR PROC FILE	The specified file name is not valid for a procedure file. Copy the procedure to another file.	BATCHPRO
F	6029	ILLEGAL SET/IF OPERATOR	An illegal operator was used in a SET statement or IF condition.	BATCHPRO
F	6030	ILLEGAL JCL VARIABLE NAME	An illegal name was used for a control statement variable in an IF condition.	BATCHPRO
F	6031	ILLEGAL SET/IF VALUE	An illegal value was used in a SET statement or IF condition.	BATCHPRO
F	6032	ILLEGAL IF SYNTAX	Check the syntax of your IF statement.	BATCHPRO
F	6033	ILLEGAL ELSE SYNTAX	Check the syntax of your ELSE statement.	BATCHPRO
F	6034	ILLEGAL ENDIF SYNTAX	Check the syntax of your ENDIF statement.	BATCHPRO
F	6035	ELSE/ENDIF WITH NO IF	An ELSE/ENDIF statement was found and there was no preceding IF statement.	BATCHPRO
F	6036	ELSE/ENDIF LABEL NOT FOUND	BATCHPRO could not find the ELSE or ENDIF statement when scanning for conditions, or could not find the ENDIF statement when skipping from an ELSE statement.	BATCHPRO
F	6037	LABEL/STRING TOO LONG	The LABEL/STRING is more than eight characters.	BATCHPRO
F	6970	INVALID ACCOUNT IDENTIFIER	The account identifier specified is not valid. Correct the account identifier.	CHARGE

Table B-2. System Utility Error Messages (Sheet 49 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	6971	UTILITY MUST BE PRIVILEGED	The CHARGE utility does not have privileged status. Notify a systems analyst.	CHARGE
F	6972	INTERNAL CALL TO USER VALIDATION FAILED	Notify a systems analyst.	CHARGE
F	6973	PARAMETER OR FORMAT ERROR	Correct the control statement and reexecute.	CHARGE
F	7001	UNSATISFIED EXTERNAL name	The controllee called a dynamic external that cannot be found.	LINKER
F	7002	SHORT COMMON BLOCK name IN MODULE-LENGTH nn RETAINED	The module has a length for a common block name different than nn. The first length still holds. If none in block, no message is issued.	LINKER
F	7003	DUPLICATE ENTRY name IN module-ENTRY IN FIRST USED	The module was loaded and it has a duplicate entry name. The previous entry name in the module will be used for satisfying calls.	LINKER
F	7004	UNSHARABLE LOADER TEXT IN module	Module has loader text that is not valid in a dynamic environment.	LINKER SLGEN
F	7005	INVALID LOADER TEXT FORMAT IN module	The format of the loader text for module module is not correct.	LINKER SLGEN
F	7100	FORMAT OF SHARED LIBRARY IS NOT VALID	The old shared library file format is not correct.	SLGEN
F	7101	FORMAT OF UTILITY FILE filename IS BAD	LOAD has produced a bad utility.	SLGEN
F	7103	LINKER IS NOT THE FIRST UTILITY	The first utility directive must be LINKER. This directive must also be included when no old library is specified and it must precede the LIB directive.	SLGEN
W	7104	UTILITY name WAS ADDED BEFORE SYSLIB	The LIB,SYSLIB directive must come before the utility name directive.	SLGEN
F	7105	VERSION CANNOT EXCEED 8 CHARACTERS	Correct the shared library version to be eight characters or less.	SLGEN
F	7106	LINKER UTILITY IS NOT VALID	LOAD has produced an incorrect linker utility.	SLGEN

Table B-2. System Utility Error Messages (Sheet 50 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	7107	DIRECTIVE SPECIFIED IS NOT VALID	A directive to SLGEN is incorrect. Refer to the SLGEN description in the VSOS Installation Handbook.	SLGEN
F	7108	DIRECTIVE EXCEEDS MAXIMUM DIRECTIVE LENGTH	A directive to SLGEN is too long to be processed. Refer to the SLGEN description in the VSOS Installation Handbook.	SLGEN
F	7109	FILENAME EXCEEDS MAXIMUM FILE LENGTH	The maximum length for a file name is eight characters or less.	SLGEN
F	7110	WRONG FORMAT USED SPECIFYING DIRECTIVES	Refer to chapter 4 of this manual for the correct format of directives in this manual.	SLGEN
F	7111	EOF REACHED BEFORE END DIRECTIVE	Add the END directive to your file.	SLGEN
F	7112	LINKER ALREADY EXISTS ON LIBRARY	A directive is trying to add a linker utility, and one already exists.	SLGEN
F	7113	INVALID LIBRARY libfile	The library specified by the LIB directive is incorrect.	SLGEN
F	7114	ORIGIN IS MUTUALLY EXCLUSIVE WITH OLDLIB	The OLDLIB is already fixed for some addresses that cannot be changed.	SLGEN
F	7200	FILE filename IS NOT EXECUTABLE	The file specified to TASKATT must be a controllee file.	TASKATT
F	7201	FILE filename IS NOT A DYNAMIC/SHARED LIBRARY	The file specified by the ULIB parameter is not a dynamic or shared library.	TASKATT
F	7202	CONTROLLEE filename WAS NOT LOADED WITH THE SYSTEM SHARED LIBRARY	The SLIB parameter was specified for a controllee that was not loaded to use a system shared library.	TASKATT
F	7203	CONTROLLEE filename WAS NOT LOADED WITH A USER DYNAMIC LIBRARY	The ULIB parameter was specified for a controllee that was not loaded to use a user dynamic library.	TASKATT
F	7204	CONTROLLEE filename WAS LOADED TO USE LIBRARY STATICLY	Neither ULIB nor SLIB can be specified for a controller which was loaded to use a library statically. There would be no assurance that the new library entry points are located at the same locations as the entry points in the original library.	TASKATT
F	7205	FILE filename DOES NOT EXIST OR IS NOT ATTACHED	The named controllee or library file either does not exist or is not attached.	TASKATT

Table B-2. System Utility Error Messages (Sheet 51 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	7206	CAN NOT BE USED TO MODIFY A DROP FILE	TASKATT can not be used to alter attributes of a drop file.	TASKATT
F	7250	DLIB REQUIRED WHEN ORIGIN IS SPECIFIED	Origin can be specified only for dynamic libraries.	OLE
W	7251	MAXIMUM ORIGIN ADDRESS MUST BE nn TO USE DEBUG	The origin address plus the length of the library conflicts with DEBUG.	OLE
F	7252	INPUT FILE filename WAS BUILT WITH DLIB	A library built with the DLIB option may not be used as input to OLE.	OLE
F	7300	FILE filename IS NOT A DROP FILE OR CONTROLLEE	The file specified to DUMP must be a drop file or a controllee file.	DUMP
W	7301	FILE filename IS NOT A SHARED LIBRARY	The drop file incorrectly references a file as a shared library.	DUMP
F	7302	NO FILE NAME SPECIFIED	A filename must be specified to DUMP.	DUMP
W	7303	FILE filename IS A BAD DROP FILE OR BAD CONTROLLEE	DUMP is unable to process completely the file specified.	DUMP
F	7304	DUMP BUG - DPRINT FAILED	DUMP logic problem. Notify a systems analyst.	DUMP
F	7305	DUMP BUG - ASCII_16 FAILED	DUMP logic problem. Notify a systems analyst.	DUMP
F	7306	ILLEGAL PARAMETER: parameter	DUMP logic problem. Notify a systems analyst.	DUMP
W	7307	CONTROLLEE WAS NOT LOADED WITH SHARED LIB SHRLIB	This is a warning indicating that the controllee may not run correctly with the shared library it is using.	DUMP DEBUG
W	7308	CONTROLLEE WAS NOT LOADED WITH DYNAMIC LIB filename	This is a warning indicating the controllee may not run correctly with the user dynamic library it is using.	DEBUG DUMP
W	7309	TOO MANY BREAKPOINTS/ CHANGES TO SYSTEM SHARED LIBRARY	DEBUG allows only 20 breakpoints and/or changes to the system shared library, and the operating system allows only so many changes to the shared library for all active DEBUG users. Either remove some of your breakpoints/changes or wait until there are fewer active DEBUG users.	DEBUG

Table B-2. System Utility Error Messages (Sheet 52 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
W	7310	SHARED LIBRARY WORKING SET IS TOO SMALL FOR BREAKPOINTS/CHANGES	The working set for the system shared library is too small to allow any more breakpoints or changes; ask operator to increase the working set or remove some breakpoints or changes.	DEBUG
W	7311	DUMP BUG-DUMP TABLE OVERFLOW	Dump had overflowed its table. Notify a systems analyst.	DUMP
W	7312	LOCATION IN SHARED LIBRARY IS ALREADY ALTERED	You can not alter the same location twice without restoring the first change.	DEBUG
W	7400	FID IS LONGER THAN 17 CHARACTERS	FID will be truncated to 17 characters.	LABEL
W	8001	INVALID COMMAND command RECEIVED	RHF protocol error; notify a systems analyst.	ACFETCH ACSTORE
W	8002	INVALID ATTRIBUTE RECEIVED attribute	RHF protocol error; notify a systems analyst.	APFETCH APSTORE RF_KATR RF_MAO1 RF_MAO2 RF_MA11 RF_MA13 RF_MA14 RF_MA15 RF_MA18 RF_MA19 RF_PA01 RF_PA02 RF_PA11 RF-PA12 RF_PA13 RF_PA14 RF_PA15 RF_PA18 RF_PA19
W	8003	QUALIFIER IS NOT I, S, OR M, QULAIPIER IS: qualifier	RHF protocol error; notify a systems analyst.	APSTORE RF_PA12

Table B-2. System Utility Error Messages (Sheet 53 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
W	8004	INVALID TEXTL. TEXTL IS character length BUT SHOULD BE character length	RHF protocol error; notify a systems analyst.	APSTORE RF_PA00 RF_PA03 RF_PA04 RF_PA05 RF_PA06 RF_PA07 RF_PA08 RF_PA12 RF_PA16 RF_PA17 RF_PA20 RF_PA21 RF_PA22 RF_PA23 RF_PA24 RF_PA26 RF_PA27 RF_PA28 RF_PA32 RF_PA33 RF_PA34
F	8005	LID MUST BE 3 ALPHA-NUM CHARS. LID IS lid	The logical identifier specified on the ST=parameter must be three alphabetic characters.	MFLINK MFQUEUE SUBMIT RF_PA25 PFUTIL
F	8006	JCS AND INPUT ARE MUTUALLY EXCLUSIVE	The user cannot specify both the JCS and the I parameter.	MFLINK MFQUEUE SUBMIT
F	8007	SYSTEM ERROR xxxxxxxx ENCOUNTERED AT HEX BIT ADDRESS aaaaaaaaaaaaaaaaaa	The task aborted due to a system problem and is terminating under Abnormal Termination Control (ATC). System errors are documented in appendix B of the VSOS Reference Manual, Volume 2.	RF_RPRVE RF_RPV
I	8008	COMMAND BUFFER FULL, CONTINUATION ATTRIBUTE INSERTED	Warning message; the set of user control statements is unusually long.	APSTORE
I	8009	DIR MUST BE I OR M	RHF protocol error; notify a systems analyst.	RF_MDBC
W	8010	PRU, EOI, EOR, AND RES MUST BE 0 OR 1	RHF protocol error; notify a systems analyst.	RF_MDBC

Table B-2. System Utility Error Messages (Sheet 54 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
W	8011	LEVEL MUST BE BETWEEN 0 AND #F	RHF encountered a file structure indicator whose record level was not in the range 0 to #F. Check that the correct data format declaration is specified.	RF_MDBC
F	8012	DD MUST BE C6 OR C8 DD is: xxxx	The specified DD= parameter must be two characters. Refer to description of MFQUEUE in chapter 3 of this manual.	RF_PA12 RF_PA31 RF_RZB
F	8013	REMOTE HOST COMMAND SEQUENCE ERR. SENT COMMAND command. RVCD command	RHF software error; notify a systems analyst.	RF_CMCRK
F	8014	REMOTE HOST ATTRIBUTE SEQUENCE ERROR	RHF software error; notify a systems analyst.	RF_CMCRK RF_EACB
F	8015	directive DIRECTIVE HAS SYNTAX ERROR	Self-explanatory. Correct error and retry.	PTFS
F	8016	RHF INTERNAL LOGIC ERROR ENCOUNTERED - xxxxxxxxx...x	RHF software error; notify a systems analyst.	DUMPF LOADPF PFERROR PTFS RF_CMBLD RF_DC8J RF_EACB RF_INTC RF_MA10 RF_PA12 RF_PTRF RF_RECOV RF_RESFN RF_SRP RF_TERM
F	8017	EXPECTED CONTINUATION BLOCK WAS NOT RECEIVED	Data transfer error. This error could be due to unusually long control statements.	RF_EACB
F	8018	ILLEGAL COMMAND RECEIVED FROM REMOTE HOST. COMMAND WAS command	RHF protocol error; notify a systems analyst.	RF_CMCRK RF_EACB
F	8019	ILLEGAL ATTRIBUTE attribute RECEIVED ON COMMAND command	RHF protocol error; notify a systems analyst.	RF_EACB RF_RQUIT

Table B-2. System Utility Error Messages (Sheet 55 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8020	REMOTE HOST lid REJECTED CONNECT WITH UNDEFINED REASON CODE, RC = #0	LCN software error. Inform a systems analyst.	RF_MCONN
W	8021	INVALID CHARACTER FOUND. CHAR.: character WRD NO word number CHR NO char- acter number	RHF protocol error. RHF encountered a character code not included in the ASCII character set. Its hexadecimal code was outside the range #20 to #7E. Notify a systems analyst.	APSTORE
F	8022	ABN #xxxxxxx NOT FOUND IN ABN TABLE T_OSB	RHF protocol error. Attribute block number could not be found in the attribute block number table T_OSB. Notify a systems analyst.	RF_FABN
F	8023	ILLEGAL BLOCK TYPE RECEIVED FROM REMOTE HOST - block type	RHF protocol error; notify a systems analyst.	RF_RCVBK RF_RCVCS RF_RECOV RF_VENR RF_VL6HD
I	8024	TOO MANY OUTSTANDING BLOCKS HAVE NOT BEEN BACKED	RHF protocol error; notify a systems analyst.	RF_SABN
F	8025	INVALID LFN - filename	The specified file name is invalid.	RF_SPLIT
F	8026	CONTROLLEE ABORTED	A fatal error in the MFLINK control statement sequence caused its execution abort. Ensure that the attached pool or private file does not have the same name as a control verb.	RF_INTC
F	8027	OPERATOR KILLED JOB	The OPERATOR executed an n.KILL command which caused the task to terminate without any further processing.	RF_INTC
F	8028	RCD NAD FOR LID lid IS DISABLED	The operator has logically disabled the RCD NAD(s) required to perform the MFLINK. Either wait until the operator enables the RCD NAD or choose another LID.	RF_VALID
F	8029	OPERATOR DROPPED JOB	The OPERATOR executed an n.DROP command which causes the task to terminate. Processing may continue if an EXIT statement follows in the job stream.	RF_INTC

Table B-2. System Utility Error Messages (Sheet 56 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
I	8030	CONTROLLEE TERMINATED, FILES NOT SAVED	Informative message; execution of the control statement sequence terminated without saving the files changed by the sequence.	RF_INTC
I	8031	CONTROLLEE TERMINATED, FILES SAVED	Informative message; execution of the control statement sequence terminated with the files changed by the sequence stored and available for later access.	RF_INTC
F	8032	USER ENTERED TERMINAL MESSAGE TRANSFERRING CONTROL TO EXIT CARD	The interactive user entered a break character during execution of the copied MFLINK control statement sequence. The results of the control statement execution are discarded.	RF_INTC
F	8033	SHD NAD FOR LID lid IS DISABLED	The operator has logically disabled the SHD NAD(s) required to perform the MFLINK. Either wait until the operator enables the SHD NAD or choose another LID.	RF_VALID
F	8034	NO END OF LINE CHARACTER FOUND	RHF encountered an error in the file format. Ensure that the correct data format is specified.	RF_RZB
W	8035	RECOVERY NOT IMPLEMENTED	RHF software error; notify a systems analyst.	RF_PUTXF
F	8036	BAD FILE - filename	The format of the copied file did not correspond to the specified data format declaration. Ensure that the correct DD=parameter option is specified.	RF_PUTXF
W	8037	FAILURE TO BUILD FIRST MESSAGE	RHF software error; notify a systems analyst.	RF_PUTXF
F	8038	FAILURE TO BUILD MESSAGE	RHF software error; notify a systems analyst.	RF_PUTXF
W	8039	DATA MESSAGE REJECTED	System software error; notify a systems analyst.	RF_PUTXF
W	8040	EOI MESSAGE REJECTED	RHF protocol error; notify a systems analyst.	RF_PUTXF
F	8041	TRANSFER HALTED BY GET DATA MESSAGE COLLAPSE	RHF software error; notify a systems analyst.	RF_BLCND

Table B-2. System Utility Error Messages (Sheet 57 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8042	TRANSFER HALTED BY BUILD DATA MESSAGE COLLAPSE	RHF software error; notify a systems analyst.	RF_BLCND
F	8048	FAMILY MEMBER TRANSITION HALTED. FAILED TO RELEASE EXISTING MEMBER	RHF software error; notify a systems analyst.	RS_TBFM
F	8049	FAMILY MEMBER TRANSITION HALTED. FAILED TO ACQUIRE NEXT MEMBER	RHF software error; notify a systems analyst.	RS_TBFM
F	8050	MAX CONTROL STATEMENTS EXCEEDED	The number of control statements specified on the MFLINK or MFQUEUE statement exceeded the maximum limit. Up to ten statements can be specified on a MFLINK statement; up to three statements on an MFQUEUE statement.	RF_PA05 RF_STORT
F	8051	DD MUST BE 2 CHARS. DD = character length CHARS	The specified DD= parameter option must be two characters.	RF_PA31
F	8052	INVALID KEYWORD keyword IN CONTROL STATEMENT, APPLICATION IS application	The RHF application did not recognize the indicated parameter keyword in the control statement. Check the control statement format for the valid parameters.	RF_PA05
F	8053	NETON NOT PREVIOUSLY CALLED BEFORE - NETXXXX CALL	RHF software error; notify a systems analyst.	NETGET NETPUT NETWAIT
F	8054	BLOCK COUNT ERROR ON - NETXXXX CALL #00000000	RHF protocol error; notify a systems analyst.	NETGET NETPUT
F	8055	INVALID BLOCK TYPE IN NETXXXXX CALL #00000000	RHF protocol error; notify a systems analyst.	NETPUT
W	8056	PHYSICAL BLOCK block ADDR= address WAS NOT LOCKED DOWN	RHF protocol error; notify a systems analyst.	NETWAIT
F	8057	APPLICATION xxxxxxxx WAS NOT INSTALLED AS A PRIVILEGED TASK	Installation error. The application was not installed in the system pool as a privileged utility.	MFLINK MFQUEUE PFUSER
I	8058	NEITHER IMPLICIT OR EXPLICIT TEXT EXIST	Informative message; applies only to QTF when sending files to a remote host.	RF_MA05
F	8061	APPLICATION IS NOT ALLOWED TO RUN ON THIS USER NUMBER: user number	Self-explanatory; if problem not caused by a user, notify a systems analyst.	RF_INTQ

Table B-2. System Utility Error Messages (Sheet 59 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8062	ERROR FROM #2A CALL OPT #6, RC = #69 - NO EMPTY SLOT FOR T CAT ORDINAL IN THE APPLICATION ENTRY IN T_CRT	The system may be busy, or a possible configuration problem exists, or an RHF software problem exists. If problem persists, notify a systems analyst.	RF_VALID
F	8063	ERROR FROM #2A CALL OPT #6, RC = #6A - NO EMPTY ENTRY IN T_CAT	The system may be busy, or a possible configuration problem exists, or an RHF software problem exists. If problem persists, notify a systems analyst.	RF_VALID
I	8064	EXCEEDED CURRENT MAXIMUM ECHO TEXT LENGTH OF 256. LENGTH IS: length	A minor RHF protocol violation has occurred; notify a systems analyst.	RF_PA29
F	8065	filename -S RT=record type BUT THATS INCOMPATIBLE WITH THE DD OF data declaration	The file has a record type which cannot be used with the specified data declaration. Either change the data declaration or the file's record type.	RF_AQXF RF_MFQ RF_OPYF
F	8066	USER CONTROL STATEMENT FORMATTED INCORRECTLY	The USER control statement has an error in its syntax. Check the USER control statement format description.	RF_PA05
F	8067	BAD USER NUMBER, ACCOUNT, PASSWORD, OR SECURITY LEVEL	At least one of the USER validation parameters is invalid. Correct the statement and retry.	RF_VUSER
F	8068	RECEIVED UNEXPECTED ERROR ON VALIDATION OF USER CARD FROM #23 OPT 6 SYSTEM MESSAGE. SS=response code	Check VSOS Reference Manual, Volume 2, system message call #23 for SS response code. Notify a systems analyst.	RF_VUSER
F	8069	NO USER CONTROL STATEMENT SPECIFIED	A USER control statement is required. Specify a USER statement as the first control statement sent.	RF_PA05
F	8070	xxxxxxx IS NOT A PUBLIC OR SYSTEM POOL CONTROLLEE	Either xxxxxxxx is a private copy of a valid controllee or it is nonexistent. If nonexistent, notify a systems analyst.	RF_TPEF
F	8071	COULD NOT SEND A QUIT TO REMOTE HOST WITH LID=lid	RHF protocol error; notify a systems analyst.	RF_SQUIT
F	8072	ATTEMPT TO VALIDATE CONTENTS OF LAST GROUP FILE FAILED	RHF software error. Inform a systems analyst.	RF_RLGF

Table B-2. System Utility Error Messages (Sheet 58 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8073	FILE: xxxxxxxx IS EMPTY. FILE NOT MFQUEUED	User attempted to MFQUEUE an empty file.	RF_MFQ
I	8074	PURGING LONE LAST GROUP FILE xxxxxxxx	Informative message for system dayfile.	RF_FCOLL
F	8075	CURRENT BLOCK COUNT ISSUED xxxx DOES NOT MATCH CURRENT BLOCK COUNT RETURNED yyyy	RHF software error; notify a systems analyst.	RF_EXNR RF_VENR RF_VL6HD
F	8076	NO BLOCKS TRANSFERRED BY NAD	RHF software error; notify a systems analyst.	RF_EXNR RF_VENR
F	8078	A LEVEL 7 PROTOCOL ERROR err WAS IGNORED WHILE BUILDING ATTRIBUTE xxxx	RHF software error; notify a systems analyst.	RF_IACB
F	8079	ILLEGAL ATTRIBUTE FOR APPLICATION	LCN level 7 protocol error; notify a systems analyst.	RF_MAO5 RF_MAI7 RF_PAO5
I	8080	FILES filename1 AND filename2 GIVEN TO USER 6	Informative message for system dayfile.	RF_MFQ
F	8081	jcs IS NOT AN ALLOWED JOB CONTROL STATEMENT	User should correct problem and retry.	RF_PAO5
F	8082	SIL ERROR OCCURRED WHILE ATTEMPTING TO REPORT USER STATEMENT ERROR	VSOS software error; notify a systems analyst.	RF_RSTOP
F	8083	SYNTAX ERROR IN USER STATEMENT	User should correct error and retry.	RF_RSTOP
F	8084	TOO MANY BLOCKS RECEIVED	LCN level 6 protocol error; notify a systems analyst.	RF_VENR
F	8085	BAD DBC dbc RECEIVED	LCN level 6 protocol error; notify a systems analyst.	RF_VENR
F	8086	DATA MESSAGE NO LONGER INDIRECT	RHF software error; notify a systems analyst.	RF_VENR
F	8087	WRONG AMOUNT OF DATA RECEIVED, xxx BITS, EXPECTED yyyy BITS	RHF software error; notify a systems analyst.	RF_VL6HD
F	8088	DID NOT RECEIVE BYTE MULTIPLE FOR CODED FILE, RECEIVED xxxx BITS	RHF software error; notify a systems analyst.	RF_VL6HD

Table B-2. System Utility Error Messages (Sheet 60 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8089	ABN WAS OUT OF SEQUENCE, RECEIVED xxxx, EXPECTED yyyy	RHF software error; notify a systems analyst.	RF_VL6HD
F	8090	RECEIVED DATA AFTER EOI	RHF software error; notify a systems analyst.	RF_VL6HD
F	8091	FILE lfn DOES NOT HAVE READ ACCESS PERMISSION	VSOS file lfn cannot be used for the transfer because this user does not have read permission for the file.	RF_AQXF
F	8092	SECOND COMMAND FOUND	RHF software error; notify a systems analyst.	RF_VL6HD
F	8093	EOI AND COMMAND FOUND	RHF software error; notify a systems analyst.	RF_VL6HD
F	8094	EOR OR EOI NOT SET IN DBC FOR LESS THAN FULL DATA BLOCK	LCN level 6 protocol error; notify a systems analyst.	RF_DUUU
W	8095	REGISTER #FC=#xxxx AFTER DIRECT KERNEL CALL #yyyy	VSOS software error; notify a systems analyst.	RF_7XC RF_GVUP RF_LKPIN RF_UNLKP RF_VPAMP
F	8096	AN ERROR WAS RETURNED FOR PARAMETER SET xxxx AFTER DIRECT KERNEL CALL #yyyy	RHF software error; notify a systems analyst.	RF_TCRCE
F	8097	INVALID VALUE wwwwww IN LOCAL VAR xxxxxxxx, VALID RANGE = yyyyyyy	An invalid value of wwwwww was encountered in the local variable named xxxxxxxx. The valid range of values for this variable is yyyyyyy. This is an RHF software error; notify a systems analyst.	RF_ERROR PFERROR
F	8098	INVALID VALUE wwwwww IN COM BLK VAR xxxxxxxx, VALID RANGE = yyyyyyy, CALLED BY - zzzzzzz	An invalid of wwwwww was encountered in the common block variable named xxxxxxxx. The valid range of values for this variable is yyyyyyy. The routine issuing the message was called by routine zzzzzzz. This is an RHF software error; contact a systems analyst.	RF_PSERR RF_PSEV RF_TERM LOADPF

Table B-2. System Utility Error Messages (Sheet 61 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8099	INVALID VALUE wwwwww IN ENTRY PARM xxxxxxxx, VALID RANGE = yyyyyyy, CALLED BY - zzzzzzz	An invalid value of wwwwww was encountered in the entry parameter named xxxxxxxx. The valid range of values for this variable is yyyyyyy. The routine issuing the message was called by routine zzzzzzz. This is an RHF software error; contact a systems analyst.	RF_PSEV RF_PRTXT PFLISTOP PFORMAT
F	8100	CONNECTED ON PATH ZERO	RHF software error; notify a systems analyst.	RF_SQTF5
F	8101	ROUTING ERROR - FROM #C70x CALL, RC=#02	RHF software or configuration error; notify a systems analyst. If the problem is a configuration error, the connection may be using a TCU that is not connected to a trunk.	RF_EXNR RF_MCONN RF_PRGP RF_RCVBK RF_RCVCS RF_RNDS RF_SNDCS
F	8102	PATH ABORTED BY REMOTE NAD - FROM #C70x CALL, RC=#03	LCN hardware/software error. Retry, and if problem persists, notify systems analyst. routine is the name of the routine that caused the problem.	RF_EXNR RF_MCONN RF_PRGP RF_RCVBK RF_RCVCS RF_RNDS RF_SNDCS
F	8103	ILLEGAL PATH NUMBER - FROM #C70x CALL, RC=#04	RHF software error; notify a systems analyst.	RF_EXNR RF_MCONN RF_PRGP RF_RCVBK RF_RCVCS RF_RNDS RF_SNDCS
F	8104	NO PATHS AVAILABLE - FROM #C70x CALL, RC=#05	Informative message. The RHF software is attempting to connect to the remote host, but the RCD NAD is so busy it doesn't have any paths to allocate to the connection. The RHF software will automatically try again after at least 10 seconds.	RF_MCONN RF_RNDS
F	8105	ILLEGAL COMMAND FOR CURRENT PATH STATE - FROM #C70x CALL RC=#06	RHF software error; notify a systems analyst.	RF_EXNR RF_MCONN RF_PRGP RF_RCVBK RF_RCVCS RF_RNDS RF_SNDCS

Table B-2. System Utility Error Messages (Sheet 62 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8106	ILLEGAL FUNCTION CODE - FROM #C70x CALL, RC=#07	RHF software error; notify a systems analyst.	RF_EXNR RF_MCONN RF_PGRP RF_RCVBK RF_RCVCS RF_RNDS RF_SNDCS
F	8107	PATH TERMINATED BY HOST REQUEST - FROM #C70x CALL, RC=#08	RHF software error; notify a systems analyst.	RF_EXNR RF_MCONN RF_PGRP RF_RCVBK RF_RCVCS RF_RNDS RF_SNDCS
F	8108	DISCONNECT MESSAGE RECEIVED - FROM #C70x CALL, RC=#09	RHF software error; notify a systems analyst.	RF_EXNR RF_MCONN RF_PGRP RF_RCVBK RF_RCVCS RF_RNDS RF_SNDCS
F	8109	CONNECT REJECTED - FROM #C70x CALL, RC=#0A	RHF software error; notify a systems analyst.	RF_EXNR RF_MCONN RF_PGRP RF_RCVBK RF_RCVCS RF_RNDS RF_SNDCS
F	8110	ILLEGAL SIZE SPECIFIED FOR MESSAGE - FROM #C70x CALL, RC=#0B	RHF software error; notify a systems analyst.	RF_EXNR RF_MCONN RF_PGRP RF_RCVBK RF_RCVCS RF_RNDS RF_SNDCS
I	8111	READ MESSAGE PRESENT - #C70x CALL, RC=#0C	The RCD NAD indicates that a message from the remote system is waiting to be read.	RF_EXNR RF_MCONN RF_PGRP RF_RCVBK RF_RCVCS RF_RNDS RF_SNDCS

Table B-2. System Utility Error Messages (Sheet 63 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8112	ILLEGAL PAGE ADDRESS - FROM #C70x CALL, RC=#0D	RHF software error; notify a systems analyst.	RF_EXNR RF_MCONN RF_PRGP RF_RCVBK RF_RCVCS RF_RNDS RF_SNDCS
F	8113	EXCESS DATA DISCARDED - FROM #C70x CALL, RC=#0E	RHF software error; notify a systems analyst.	RF_EXNR RF_MCONN RF_PRGP RF_RCVBK RF_RCVCS RF_RNDS RF_SNDCS
F	8114	INDIRECT BUFFER REQUIRED - FROM #C70x CALL, RC=#0F	RHF software error; notify a systems analyst.	RF_EXNR RF_MCONN RF_PRGP RF_RCVBK RF_RCVCS RF_RNDS RF_SNDCS
F	8115, 8118- 8141	ERROR FROM #2A CALL OPT #xx, RC=#yy - zzz...zzz	RHF software error; an error status=#yy was returned from system call #2A option #xx. The message zzz...zzz is the English description of the error; notify a systems analyst.	NETOFF NETON RF_FFF RF_INTA RF_INTP RF_INTQ RF_IRF2A RF_SQTFS RF_VALID
F	8116	UNDEFINED LOCAL SERVER APPLICATION	The specified application is not in the configuration. Notify a system's analyst.	
F	8117	APPLICATION application IS NOT IN T_CAT	The specified application is not in table T_CAT. Notify a system's analyst.	
F	8142	NO EMPTY ENTRIES IN T_CAT RHF activity, or RHF applications are aborting without removing their entry from T_CAT. Notify a system's analyst.	Either the T_CAT length was defined too small for busy.	

Table B-2. System Utility Error Messages (Sheet 64 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8143	NO EMPTY SLOT FOR T_CAT ORDINAL FOR application ENTRY IN T_CAT	Either the T_CAT length was defined too small for busy RHF activity, or RHF applications are aborting without removing their entry from T_CAT. Notify a system's analyst.	
F	8147	LID lid NOT FOUND	LID specified is not in the configuration. Error is from #2A call opt #02, rc=#21.	NETOFF NETON RF_FFF RF_INTA RF_INTP RF_INTQ RF_IRF2A RF_SQTFS RF_VALID
F	8148	LID lid IS DISABLED	An operator disabled the specified LID. Error is from #2A call opt #02, rc=#22.	NETOFF NETON RF_FFF RF_INTA RF_INTP RF_INTQ RF_IRF2A RF_SQTFS RF_VALID
F	8149	application CURRENTLY RUNNING LIMIT EXCEEDED	User should wait at least 10 seconds and try again. Error is from #2A call opt #02, rc=#23.	NETOFF NETON RF_FFF RF_INTA RF_INTP RF_INTQ RF_IRF2A RF_SQTFS RF_VALID
F	8150	RHF APPLICATION IS NOT ALLOWED TO EXECUTE ON THIS USER NUMBER	A user attempted to startup an RHF application under a user number that was not validated to run that application. Error is from #2A call opt #02, rc=#25.	NETOFF NETON RF_INTA RF_INTP RF_INTQ RF_IRF2A RF_SQTFS RF_VALID RF_FFFF

Table B-2. System Utility Error Messages (Sheet 65 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8151	RUNNING APPLICATION LIMIT EQUALS THE MAXIMUM	User should wait at least 10 seconds and try again. Error is from #2A call opt #04, rc=#01.	NETOFF NETON RF_FFF RF_INTA RF_INTP RF_INTQ RF_IRF2A RF_SQTFS RF_VALID
F	8152	RUNNING APPLICATION LIMIT EXCEEDS THE MAXIMUM	User should wait at least 10 seconds and try again. Error is from #2A call opt #04, rc=#04.	NETOFF NETON RF_FFF RF_INTA RF_INTP RF_INTQ RF_IRF2A RF_SQTFS RF_VALID
F	8153	LID NOT FOUND	User specified a LID that was not defined in the configuration. Error is from #2A call opt #06, rc=#61.	NETOFF NETON RF_FFF RF_INTA RF_INTP RF_INTQ RF_IRF2A RF_SQTFS RF_VALID
F	8154	LID lid IS DISABLED	An operator disabled the specified LID. Error is from #2A call opt #06, rc=#62.	NETOFF NETON RF_FFF RF_INTA RF_INTP RF_INTQ RF_IRF2A RF_SQTFS RF_VALID
F	8155	MAXIMUM NUMBER OF APPLICATIONS ALREADY RUNNING	Fatal informative message. Retry at least 10 seconds. Error is from #2A call opt #06, rc=#63.	RF_FFF RF_SQTFS RF_VALID
F	8156	APPLICATION IS NOT ALLOWED TO EXECUTE ON THIS USER NUMBER	An attempt was made to startup an RHF application under a user number that was not validated to run that application. Error is from #2A call opt #06, rc=#65.	NETOFF NETON RF_FFF RF_INTA RF_INTP RF_INTQ RF_IRF2A RF_SQTFS RF_VALID

Table B-2. System Utility Error Messages (Sheet 66 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8157	PID pid IS DISABLED	The operator has disabled the PID. Error is from #2A call opt #06, rc=#68.	NETOFF NETON RF_FFF RF_INTA RF_INTP RF_INTQ RF_IRF2A RF_SQTF5 RF_VALID
F	8158	LID NOT FOUND	LID specified was not defined. Error is from #2A call opt #0E, rc=#E1.	NETOFF NETON RF_FFF RF_INTA RF_INTP RF_INTQ RF_IRF2A RF_SQTF5 RF_VALID
F	8159	TRANSFER REJECTED, CAUSE UNKNOWN - FROM level 7 command	Remote application rejected file transfer without giving cause.	RF_L7QSS RF_SSTOP
F	8160	TRANSFER REJECTED, REFER TO ACCOMPANYING MESSAGE - FROM level 7 command	Remote application rejected file transfer; the reason is provided by an additional message either immediately preceding or following this message.	RF_L7QSS RF_SSTOP
F	8161	TRANSFER REJECTED, UNACCEPTABLE TRANSFER CONTROL SETTING - FROM level 7 command	Remote application rejected file transfer because one of the transfer specifications (usually explicit text) was unacceptable.	RF_L7QSS RF_SSTOP
F	8162	TRANSFER REJECTED, UNSPECIFIC FILESTORE - FROM level 7 command	The remote application rejected the file transfer because it needed more information to store the file.	RF_L7QSS RF_SSTOP
F	8163	TRANSFER REJECTED, FILE NOT FOUND OR DOES NOT EXIST - FROM level 7 command	The remote application rejected the file transfer because it could not find the file.	RF_L7QSS RF_SSTOP
F	8164	TRANSFER REJECTED, NO ACCESS TO FILE QUOTED - FROM level 7 command	The remote application rejected the file transfer because the user needed to specify access permission parameters.	RF_L7QSS RF_SSTOP

Table B-2. System Utility Error Messages (Sheet 67 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8165	TRANSFER REJECTED, WRONG FILE TYPE - FROM level 7 command	The remote application rejected the file transfer because the user specified the wrong file type.	RF_L7QSS RF_SSTOP
F	8166	TRANSFER REJECTED, FILE NOT AVAILABLE OR OFFLINE - FROM level 7 command	The remote application rejected the file transfer because the file was not available or off-line.	RF_L7QSS RF_SSTOP
F	8167	TRANSFER REJECTED, USERNAME UNKNOWN - FROM level 7 command	The remote application rejected the file transfer because the user specified an unknown username.	RF_L7QSS RF_SSTOP
F	8168	TRANSFER REJECTED, USERNAME PASSWORD NOT QUOTED OR QUOTED INCORRECTLY - FROM level 7 command	The remote application rejected the file transfer because the user specified an invalid password.	RF_L7QSS RF_SSTOP
F	8169	TRANSFER REJECTED, ACCOUNT UNKNOWN - FROM level 7 command	The remote application rejected the file transfer because the user specified an invalid account number.	RF_L7QSS RF_SSTOP
F	8170	TRANSFER REJECTED, ACCOUNT PASSWORD NOT QUOTED OR QUOTED INCORRECTLY - FROM level 7 command	The remote application rejected the file transfer because the user failed to quote the account password or quoted it incorrectly.	RF_L7QSS RF_SSTOP
F	8171	TRANSFER REJECTED, NO MONEY LEFT - FROM level 7 command	The remote application rejected the file transfer because the user's account has no money left.	RF_L7QSS RF_SSTOP
F	8172	TRANSFER REJECTED, FILE SIZE QUOTED TOO BIG - FROM level 7 command	The remote application rejected the file transfer because it does not have file space left for the file size quoted.	RF_L7QSS RF_SSTOP
F	8173	TRANSFER REJECTED, OUTPUT DEVICE UNKNOWN OR UNAVAILABLE - FROM level 7 command	The remote application rejected the file transfer because the user specified an output device that was unknown or unavailable.	RF_L7QSS RF_SSTOP
F	8174	TRANSFER ABORTED, REFER TO ACCOMPANYING MESSAGE - FROM level 7 command	The remote application terminated the file transfer abnormally; the reason is provided by an additional message immediately preceding or following this message.	RF_L7QSS RF_SSTOP
I	8175	TRANSFER TERMINATED, SATISFACTORY AND INCOMPLETE, NO RETRY REQUIRED - FROM level 7 command	The remote application terminated the file transfer prematurely but without fatal consequences.	RF_L7QSS RF_SSTOP

Table B-2. System Utility Error Messages (Sheet 68 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8176	TRANSFER ABORTED, RECEIVER PROBLEMS, RETRY POSSIBLE - FROM level 7 command	The remote application aborted the file transfer due to receiver problems; a retry may be successful.	RF_L7QSS RF_SSTOP
F	8177	TRANSFER ABORTED, RECEIVER PROBLEMS, NO RETRY POSSIBLE - FROM level 7 command	The remote application aborted the file transfer due to receiver problems; retrying will result in the same error.	RF_L7QSS RF_SSTOP
F	8178	TRANSFER ABORTED, SENDER PROBLEMS, RETRY POSSIBLE - FROM level 7 command	The remote application aborted the file transfer due to sender problems; a retry may be successful.	RF_L7QSS RF_SSTOP
F	8179	TRANSFER ABORTED, SENDER PROBLEMS, NO RETRY POSSIBLE - FROM level 7 command	The remote application aborted the file transfer due to sender problems; retrying will result in the same error.	RF_L7QSS RF_SSTOP
F	8180	TRANSFER ABORTED, TIME-OUT MATURED - FROM level 7 command	The remote application aborted the file transfer because it ran out of time.	RF_L7QSS RF_SSTOP
F	8181	TRANSFER ABORTED, IRRECOVERABLE PROTOCOL ANOMALY - FROM level 7 command	The remote application aborted the file transfer because it detected fatal problems in the level 7 protocol exchange.	RF_L7QSS RF_SSTOP
F	8182	DATA TRANSFER TERMINATED, RECEIVE ERROR, RETRY POSSIBLE - FROM level 7 command	The remote application terminated the data transfer due to receiver problems; a retry may be successful.	RF_PA23
F	8183	DATA TRANSFER TERMINATED, RECEIVE ERROR, NO RETRY POSSIBLE - FROM level 7 command	The remote application terminated the file transfer due to receiver problems; a retrying will result in the same error.	RF_PA23
F	8184	DATA TRANSFER TERMINATED, PROTOCOL ERROR DETECTED BY RECEIVER - FROM level 7 command	The remote application terminated the data transfer because it detected fatal problems in the level 7 protocol exchange.	RF_PA23
F	8185	DATA TRANSFER TERMINATED, GO NOT ACCEPTABLE TO RECEIVER - FROM level 7 command	The remote application terminated the data transfer because the receiver had fatal problems after the GO command was received.	RF_PA23
F	8186	DATA TRANSFER TERMINATED, SENDER ERROR, RETRY POSSIBLE - FROM level 7 command	The remote application terminated the data transfer because of sender problems; a retry may be successful.	RF_PA23

Table B-2. System Utility Error Messages (Sheet 69 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8187	DATA TRANSFER TERMINATED, SENDER ERROR, NO RETRY POSSIBLE - FROM level 7 command	The remote application terminated the data transfer because of sender problems; retrying will result in the same error.	RF_PA23
F	8188	DATA TRANSFER TERMINATED, PROTOCOL ERROR DETECTED BY SENDER - FROM level 7 command	The remote application terminated the data transfer because the sender detected level 7 protocol problems.	RF_PA23
F	8189	DATA TRANSFER TERMINATED, GO NOT ACCEPTABLE TO SENDER - FROM level 7 command	The remote application terminated the data transfer because the sender had problems after the GO command was received.	RF_PA23
F	8190	UNABLE TO FIND ERRCODE errcode FOR SOURCE = source - CALLED BY routine	RHF software error; notify a systems analyst.	RF_GEP
F	8191	PRIVILEGED LOAD OF NON-PRIVILEGED DUMP NOT ALLOWED	Files archived using DUMPF under a nonprivileged user number cannot be reloaded using LOADPF under a privileged user number. Reload the files using the non-privileged user number under which they were archived.	PFBUFFER
F	8192	CONNECT REQUEST SENT TO pid REJECTED - SERVER NAD DISABLED	The remote host with an ID of pid has rejected the connect request because its local RHF NAD has been logically disabled. Either wait until the remote host's operator enables the RHF NAD, or choose another LID for the MFLINK.	RF_MCONN
F	8193	CONNECT REQUEST SENT TO pid REJECTED - SERVER NAD INVALID OR UNDEFINED	The remote host with an ID of pid has rejected the connect request because the local NAD that received the request is not defined for their RHF subsystem. Contact a site analyst.	RF_MCONN
I	8194	SHD NAD #nn RETURNED PATH STATUS #0000 - PATH AVAILABLE	The RHF software is attempting to connect to a remote host, but upon checking its allocated path, it discovered that the RCD NAD had released it. The RAF software will retry the connect attempt.	RF_MCONN

Table B-2. System Utility Error Messages (Sheet 70 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8195- 8219	SHD NAD nn LID mmm RETURNED PATH STATUS #xxxx-yyy...yyy	nn is the remote NAD number. mmm is the transfer lid. #xxxx is the path status. The English description of the error is given by yyy...yyy. The path connection for this transfer is being connected or disconnected. There may be a problem with the connection. Notify a systems analyst.	RF_MCONN RF_RNDS
F	8220	CALLED WITH BAD SOURCE source FROM routine	RHF software error; notify a systems analyst.	RF_GEP
F	8221	UNABLE TO MAKE CONNECTION	The retry count for connection attempts has been exceeded; the remote system is not currently accessible.	RF_MCONN
F	8223	QUEUE FILE MUST BE LOCAL	RHF software error; notify a systems analyst.	RF_OPYF
F	8224	FILE filename DOES NOT HAVE WRITE ACCESS PERMISSION	User should use the PERMIT utility to redefine the access permission set.	RF_OPYF
F	8225	FILE filename HAS UNKNOWN RECORD TYPE rt	VSOS software error; notify a systems analyst.	RF_OPYF
I	8226	CONNECTION ESTABLISHED WITH lid	This message is issued if the RHF software was able to establish a connection to the remote host indicated by lid and if the 8300 message was previously issued.	RF_MCONN
I	8227	CONNECT REQUEST SENT TO lid REJECTED - SERVER APPLICATION yyyyyy TEMPORARILY UNAVAILABLE	Requested application has maximum number of copies running on the remote host. If interactive, retry later. If batch, initiating application will retry.	RF_MCONN
I	8228	CONNECT REQUEST SENT TO lid REJECTED - SYSTEM RESOURCES TEMPORARILY UNAVAILABLE ON REMOTE HOST	Insufficient system resources (memory, buffers or table space), on remote host to initiate the specific server application. Retry later, if interactive. If batch, initiating application will retry.	RF_MCONN

Table B-2. System Utility Error Messages (Sheet 71 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
I	8229	CONNECT REQUEST SENT TO lid REJECTED - SERVER APPLICATION yyyyyyy DISABLED	The remote host has the requested server application disabled. Retry later, if interactive. If batch, initiating application will retry.	RF_MCONN
I	8230	CONNECT REQUEST SENT TO lid REJECTED - SERVER LID DISABLED	Server LID is disabled in local tables of the remote host. If interactive, retry later. If batch, initiating application will retry.	RF_MCONN
I	8231	CONNECT REQUEST SENT TO lid REJECTED - INITIATOR APPLICATION yyyyyyy DISABLED	Initiator application is disabled in local tables of the remote host. If interactive, retry later. If batch, initiating application will retry.	RF_MCONN
I	8232	CONNECT REQUEST SENT TO lid REJECTED - INITIATOR PID DISABLED	Initiator PID is disabled in local tables of the remote host. If interactive, retry later. If batch, initiating application will retry.	RF_MCONN
I	8233	CONNECT REQUEST SENT TO lid REJECTED - INITIATOR NAD DISABLED	Initiator NAD is disabled in local tables of the remote host. If interactive, retry later. If batch, initiating application will retry.	RF_MCONN
I	8234	CONNECT REQUEST SENT TO lid REJECTED - TCU DISABLED	TCU is disabled in local tables of the remote host. If interactive, retry later. If batch, initiating application will retry.	RF_MCONN
I	8235	CONNECT REQUEST SENT TO lid REJECTED - RHF IS NOT ACTIVE	RHF is not active on the remote host. If interactive, retry later. If batch, initiating application will retry.	RF_MCONN
I	8236	CONNECT REQUEST SENT TO lid REJECTED - SYSTEM SHUTDOWN IN PROGRESS	Operator has issued idle (or termination) command to RHF on the remote host. If interactive, retry later. If batch, initiating application will retry.	RF_MCONN
F	8237	CONNECT REQUEST SENT TO lid REJECTED - SERVER APPLICATION yyyyyyy INVALID OR UNDEFINED	The remote host does not have the requested server application in its local tables. Notify a systems analyst.	RF_MCONN
F	8238	CONNECT REQUEST SENT TO lid REJECTED - SERVER LID INVALID OR UNDEFINED	Unknown server LID on the remote host. Notify a systems analyst.	RF_MCONN

Table B-2. System Utility Error Messages (Sheet 72 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8239	CONNECT REQUEST SENT TO lid REJECTED - INITIATOR APPLICATION yyyyyyy INVALID OR UNDEFINED	Unknown initiator application on the remote host. Notify a systems analyst.	RF_MCONN
F	8240	CONNECT REQUEST SENT TO lid REJECTED - INITIATOR PID INVALID OR UNDEFINED	Unknown initiator PID on the remote host. Notify a systems analyst.	RF_MCONN
F	8241	CONNECT REQUEST SENT TO lid REJECTED - INITIATOR PASSWORD INVALID OR UNDEFINED	Unknown password for requested initiator-server combination on the remote host. Notify a systems analyst.	RF_MCONN
F	8242	CONNECT REQUEST SENT TO lid REJECTED - INITIATOR NAD INVALID OR UNDEFINED	Unknown initiator NAD on the remote host. Notify a systems analyst.	RF_MCONN
F	8243	CONNECT REQUEST SENT TO lid REJECTED - ACCESS CODE INVALID OR UNDEFINED	Unknown access code for remote NAD on the remote host. Notify a systems analyst.	RF_MCONN
F	8244	CONNECT REQUEST SENT TO lid REJECTED - DESTINATION DEVICE INVALID OR UNDEFINED	Unknown destination device on the remote host. Notify a systems analyst.	RF_MCONN
F	8245	CONNECT REQUEST SENT TO lid REJECTED - TCU INVALID OR UNDEFINED	Unknown TCU on the remote host. Notify a systems analyst.	RF_MCONN
F	8275	INPUT QUEUE ON xxxxxxxx IS FULL - RESEND JOB jjjjjjjj	Input queue already has the maximum number of jobs. Job should be resubmitted at a later time.	RF_RSTOP
F	8276	INPUT QUEUE ON xxxxxxxx HAS MAX JOBS FOR THIS USER - RESEND JOB jjjjjjjj	User number specified by user card for this job already has maximum number of jobs allowed in the input queue. Job should be resubmitted at a later time.	RF_RSTOP
I	8277	FULL RECOVERY INITIATED	Connection to linked remote host was lost. Recovery attempted from beginning of request.	RF_RECOV
F	8278	RECOVERY LIMIT EXCEEDED- RECOVERY ABANDONED	The installation limit of recovery attempts was exhausted before successful completion of the transfer.	RF_RECOV

Table B-2. System Utility Error Messages (Sheet 73 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
I	8279	OPERATOR SELECTED WAIT OPTION	During RHF recovery processing, the operator selected the option to wait for the requested resources.	RF_RECOV
F	8280	OPERATOR ABORTED TASK	During RHF recovery processing, the operator selected the option to abort this task.	RF_RECOV
I	8281	OPERATOR SELECTED RETRY OPTION	During RHF recovery processing, the operator selected the option to retry this request.	RF_RECOV
F	8282	QUALIFIER IS qualifier ₁ QUALIFIER OF qualifier ₂ EXPECTED	The incoming command contained an inappropriate qualifier. LCN level 7 protocol error; notify a systems analyst.	RF_PA12
F	8283	SECONDARY USER CARDS NOT ALLOWED	PTFS with recovery text option does not support more than one user card in a session.	RF_PA05
I	8285	THE TEXT OF AN ATTRIBUTE RECEIVED ON AN RNEG HAD LENGTH xxxxxxxx SO IT WAS ADJUSTED TO 104	QTF is attempting to save the rejected parameter of a level 7 LCN RFT command for possible debug analysis later. However, the text is longer than the norm, so only the first 104 characters are saved. If the problem was not caused by a user error, notify a systems analyst.	RF_SRP
I	8286	RAN OUT OF ROOM IN REJ_PARM_BUFF BEFORE ALL THE REJECTED PARAMETERS WERE PROCESSED	QTF is attempting to save the rejected parameters of a level 7 LCN RFT command for possible debug analysis later. However, there was not enough space in the buffer REJ_PARM_BUFF to save all the text. If the problem was not caused by a user error, notify a systems analyst.	RF_SRP
I	8287	REMOTE JOBNAME xxxxxxxx CHANGED TO yyyyyyy	The remote system has sent a job or file transfer request that has jobname xxxxxxxx on that system. The equivalent jobname on the local system for the job will be yyyyyyy.	RF_PA26

Table B-2. System Utility Error Messages (Sheet 74 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8288	FILENAME MUST BE SPECIFIED FOR DATA TRANSFER	The job control statements sent to the remote system specified that a file was to be transferred, but the local file name was not entered on the execute line.	RF_L7QSS
F	8289	APPLICATION NAME MUST BE QTF OR QTFS	A routine other than QTF or QTFS attempted to make a filename attribute. Notify a systems analyst.	RF_MA16
F	8290	ERROR IN DATA TRANSFER	An error occurred during the file transfer. Transfer aborted.	PTF
F	8291	TRANSFER ENCOUNTERED NETWORK ERROR (QUIT RECEIVED)	An error was encountered during the data transfer, and a QUIT was received from the remote host.	RF_L7QFS
I	8292	BEGIN appl	Application appl was entered.	RF_Q5GTN
F	8294	ERROR RETURNED BY Q7PROMPT/Q7KEYWRD, RESPONSE CODE: xxxx	Q7PROMPT or Q7KEYWORD returned a nonzero response code of xxxx. Recheck parameter list.	MFQUEUE SUBMIT
F	8295	BAD LHS TABLE NUMBER RETURNED BY Q7PROMPT/Q7KEYWRD, TABLE #: xxx	Q7PROMPT or Q7KEYWORD returned a left-hand-side table number that was outside the range of legal left-hand-side tables. Contact a systems analyst.	MFQUEUE SUBMIT
W	8296	LID xxx IS CURRENTLY DISABLED	The LID xxx is currently disabled.	SUBMIT MFQUEUE
F	8297	LID xxx NOT FOUND	The LID xxx is not a defined LID.	SUBMIT MFQUEUE
I	8298	WAITING FOR PATH AVAILABILITY	No paths are available at this time. Controllee will wait for a path to become available.	RF_MCONN
I	8299	CONNECT PATH IS: xxx	The particular path this transfer connected to is xxx.	RF_MCONN
I	8300	WAITING TO CONNECT TO lid	The RHF software is experiencing a delay in connecting to the remote host indicated by lid.	RF_MCONN
F	8302	FILE filename IS NOT ATTACHED OR DOES NOT EXIST	The input file, filename, is not attached or does not exist. This is the file containing jcs to be sent to the remote system.	RF_OISB

Table B-2. System Utility Error Messages (Sheet 75 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8303	NO ANSWER, NAD #xx IS NOT ON THE TRUNK USED, RC=#0	The system to which the user sent an RHF statement has not responded.	RF_RNDS
F	8304	LOCAL TCI/TCU HARDWARE PROBLEM FROM NAD #xx, RC=#1	RHF hardware error; notify a systems analyst.	RF_RNDS
F	8305	RECEIVED GARBLED RESPONSE FROM REMOTE NAD #xx, RC=#2	RHF hardware error; retry and if problem persists, notify a systems analyst.	RF_RNDS
F	8306	REMOTE NAD #xx NOT RUNNING, RC=#3	RHF hardware error; can't communicate because NAD has stopped. Retry and if problem persists, notify a systems analyst.	RF_RNDS
F	8307	NO RESPONSE FROM REMOTE NAD #xx, RC=#4	RHF hardware error; response received from remote, but it was generated by remote trunk hardware (TCU) after remote NAD did not supply response message within expected time period. Retry and if problem persists, notify a systems analyst.	RF_RNDS
F	8308	LOST COMMUNICATIONS TO LONG HAUL NAD #xx, RC=#8	RHF hardware error; retry and if problem persists, notify a systems analyst.	RF_RNDS
F	8309	CAN'T INITIALIZE WITH LONG HAUL NAD #xx, RC=#9	RHF hardware error; retry and if problem persists, notify a systems analyst.	RF_RNDS
F	8310	REMOTE NAD #xx AUTO LOADED, RC=#16	NAD down on other end. Retry and if problem persists, notify a systems analyst.	RF_RNDS
F	8311	NO PATH AVAILABLE AT REMOTE NAD #xx, RC=#17	Remote NAD has no paths left; resources are all used up. Retry and if problem persists, notify a systems analyst.	RF_RNDS
F	8312	TCU SPECIFIED FOR REMOTE NAD #xx DOES NOT EXIST, RC=#18	RHF software error or possible configuration error. Notify a systems analyst.	RF_RNDS
F	8313	PATH TO REMOTE NAD #xx NO LONGER EXISTS, RC=#19	RHF software error; remote host removed path at remote end.	RF_RNDS

Table B-2. System Utility Error Messages (Sheet 76 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8314	HOST FOR REMOTE NAD #xx IS INACTIVE, RC=#20	RHF hardware error; remote host does not appear to be running to remote NAD. Host has not communicated with NAD for last 1.5 minutes. Retry and if problem persists, notify a systems analyst.	RF_RNDS
F	8315	REQUESTED APPLICATION UNAVAILABLE ON LID lid, RC=#1	Requested application on remote system is unavailable or undefined.	RF_MCONN
F	8316	RHF SHUTDOWN IN PROGRESS ON LID lid, RC=#2	Remote RHF host is down or temporarily not accepting new connections.	RF_MCONN
F	8317	INITIATOR NAD OR DESTINATION INVALID ON LID lid, RC=#3	RHF software error or possible configuration error. Retry and if problem persists, notify a systems analyst.	RE_MCONN
F	8318	PID/LID NOT AVAILABLE ON LID lid, RC=#4	Remote PID/LID has been disabled or not defined. Possible configuration error.	RF_MCONN
F	8319	INVALID PASSWORD/ACCESS CODE ON LID lid, RC=#5	RHF software error; notify a systems analyst. Possible configuration error.	RF_MCONN
F	8320	PATH OR NAD UNAVAILABLE ON LID lid, RC=#8	RHF software error; notify a systems analyst. Possible configuration error.	RF_MCONN
F	8321	INVALID PASSWORD/ACCESS CODE ON LID lid, RC=#9	RHF software error; notify a systems analyst. Possible configuration error.	RF_MCONN
F	8322	REQUESTED APPLICATION NAME UNKNOWN/INVALID ON LID lid, RC=#10	Remote application specified is not in configuration.	RF_MCONN
F	8323	RESOURCES NOT AVAILABLE ON LID lid, RC=#11	Remote RHF host does not currently have resources to accept this transfer. Retry and if problem persists, notify a systems analyst.	RF_MCONN
F	8325	RCD NAD NAD xxxxx ZIP CODE IS ZERO - CONFIGURATION PROBLEM EXISTS	System was configured incorrectly; notify a systems analyst.	RF_MCONN
I	8326	PRE 2.2 FILE FAMILY xxxxxxxx GIVEN TO USER yyyyyyyy	QTF could not transfer output file xxxxxxxx to the remote host, so it gave it to user yyyyyyyy.	RF_GFF

Table B-2. System Utility Error Messages (Sheet 77 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8327	FILE filename IS A RESERVED FILENAME	The file specified has a name reserved for special system functions and cannot be accessed by the user for transfer.	MFLINK MFQUEUE PTF PTFS SUBMIT SUBPRO RF_PA16 PFLIO PFUTIL
F	8328	FILE filename BELONGS TO A USER1 NUMBER	DUMPF or LOADPF aborted because file filename belongs to a User-1 number.	PFUTIL
F	8329	NO BUFFERS AVAILABLE FROM #C7OX CALL, RC=#10	The RHF software has encountered a busy condition in the RCD NAD. The RHF software will automatically try again after at least 10 seconds.	RF_EXNR RF_MCONN RF_PGRP RF_RCVBK RF_RCVCS RF_RNDS RF_SNDCS
F	8330	CANNOT xxxxxxxx SYSTEM USER DIRECTORY FILE	xxxxxxx is either LOADPF or DUMPF. LOADPF/DUMPF cannot load/dump the system user directory file, Q5UDF.	PFUTIL
F	8331	FILE filename IS NOT A PHYSICAL DATA FILE	The file type for the input file, filename, must be physical data. This is the file containing jcs to be sent to the remote system.	RF_OISB
F	8338	UNABLE TO FIND REASON CODE code FOR STATE CLARIFIER = #yyyyyyyy	RHF software error; notify a systems analyst.	RF_GEP
F	8339	INPUT FILE xxxxxxxx IS EMPTY	The input file xxxxxxxx specified by the user did not contain any data. Check that the proper file was specified and contains the needed control statements.	MFLINK MFQUEUE SUBMIT PFSCAN
F	8340	EITHER JCS OR INPUT MUST BE SPECIFIED	The user did not specify any control statements for the application either in the form of an explicit JCS statement or an input file. Reenter the execute line and specify one or the other.	MFLINK PFSCAN
F	8344	UNKNOWN ARCHIVE FILE FORMAT ENCOUNTERED, LRU LENGTH WAS xxxxxxxx	The user tried using LOADPF to reload files that were not archived via DUMPF.	PFLIO

Table B-2. System Utility Error Messages (Sheet 78 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8345	TAPE FORMAT MISMATCH REQUEST TF WAS xxxxxxxx, TAPE TF WAS yyyyyyyy	The user specified a tape format of xxxxxxxx when trying to reload files from a tape that was created with tape format yyyyyyyy. Execute LOADPF again specifying the correct format, yyyyyyyy.	PFLIO
F	8350	FILE NOT PROCESSED PROCESSED - NO JDNS AVAILABLE	The system has run out of available JDNS. Wait a few seconds and try again. If the problem persists, contact a site analyst.	MFQUEUE SUBMIT
F	8352	TIME LIMIT	The user's time limit to complete a task was not sufficient for the applicaton to complete. Increase the time limit specified on the RESOURCE statement and resubmit.	MFLINK MFQUEUE SUBMIT
F	8353	ILLEGAL INSTRUCTION ENCOUNTERED AT BIT ADDRESS xxxxxxxxxxxxxxxx	RHF software error or possible hardware error. Notify a systems analyst.	MFLINK MFQUEUE SUBMIT
I	8354	PROGRAM SUSPENDED FOR nnnn SECONDS - WILL RETRY AFTER RECALL	In response to a condition described in an error message displayed prior to this message, the application went into SUSPEND/RECALL for nnnn seconds; after that time, the application will retry. An interactive user may choose to break the application by entering !.	MFLINK
I	8389	ALL PSEUDO FILES ASSOCIATED WITH DVSTnn WERE PURGED	The reload is from mass storage and the user specified PURGE=YES parameter. All the user's pseudo files residing on device set DVSTnn were purged successfully.	LOADPF
F	8390	PURGE OF PSEUDO FILES FOR DVSTnn INCOMPLETE - CHECK YOUR PERMANENT FILES	The reload is from mass storage and the user specified PURGE=YES parameter. While LOADPF was purging the user's pseudo files from DVSTnn, LOADPF encountered an error (described by the previous error message). The purge of pseudo files stopped. The user should check his/her permanent files to ascertain what pseudo files did not get purged.	LOADPF

Table B-2. System Utility Error Messages (Sheet 79 of 80)

Severity	Error Code	Message	Significance	Issuing Routine
F	8391	TIMED OUT AFTER xxx SECS WAITING FOR RCD NAD ZIP = #nn TO COMPLETE FUNCTION #fff	RHF software or hardware problem. Notify a system's analyst.	NETWAIT
F	8392	PARAMETER ERROR on #FF02 CALL	RHF software problem. Notify a system's analyst.	RF_GVUP
I	8393	xxxxxx TRANSFERRED #nnnn WORDS OF DATA	The number of CYBER 200 words of data transferred to/from the remote host is #nnnn. xxxxxx is either MFGIVE or MFTAKE and indicates the direction of the transfer.	PTFS
I	8394	lid IS A LOCAL LID, DESTINATION IS CHANGED TO lid2	Informative message. The LID specified by lid1 identifies the CYBER 205 local host that is executing this application. Output will be sent to the default output remote host identified by lid2.	MFQUEUE
F	8397	SESSION RECOVERY FILE DOES NOT EXIST - RETRY SPECIFYING LID	The ST=lid parameter was not specified on the MFLINK so an attempt was made to read the session recovery file which contains the destination lid and user or accounting directives. However, the file does not exist so the user should retry specifying the ST=lid parameter.	RE_RSRF
F	8600	INVALID JOB CARD	The file being submitted did not begin with a valid job card.	SUBPRO
F	8601	INVALID USER CARD	The second control statement of the file being submitted was not a user card with a valid user number.	SUBPRO

Table B-2. System Utility Error Messages (Sheet 80 of 80)

Sever- ity	Error Code	Message	Significance	Issuing Routine
F	8603	UTILITY NOT PRIVILEGED	The SUBMIT utility has not been installed as a privileged task.	SUBMIT
F	8604	FILE MUST BE RT=R OR RT=W	The user has attempted to submit a file with a record format other than R or W.	SUBPRO
F	8605	JOB NAME MUST BE 1-8 CHARACTERS	The user attempted to submit a file with a JN parameter consisting of more than eight characters.	SUBPRO

Table B-3. System Error Codes (Sheet 1 of 2)

Hexadecimal Code	Description
5	The instruction is not in the CYBER 200 instruction set.
6	The exit force instruction does not have a pointer to a system message to be executed.
7	Illegal request.
8	Parity error in data transfer between the CPU and central memory.
9	The job is unrecoverable because of an outstanding I/O request.
A	A C50x request did not contain a file segment table ordinal.
B	Illegal C504 request.
22	An I/O error occurred for a read or write of a drop file.
24	Large page limit exceeded.
25	Page size conflict in drop file.
26	Virtual address duplicate direct fault.
27	Write violation in system call.
28	A write violation occurred while the system was swapping in a page referenced by the job.
29	The job referenced a page within the virtual system address range.
2A	The drop file map is full; the job can define no more virtual regions.
2B	The job class of the job is not allowed the use of large pages.
2C	The job referenced a page in the shared library reserved area.
2D	Drop file overflow; no more virtual space can be mapped into the drop file.
2E	The drop file map is full; no more virtual space can be mapped into the drop file.
2F	A virtual system call caused drop file overflow.

Table B-3. System Error Codes (Sheet 2 of 2)

Hexadecimal Code	Description
30	Time limit; the system allocates time for processing the interrupt subroutine.
31	The paging routine received an I/O error.
40	Bound implicit map anomaly.
51	The file segment table is full.
209	No source file.
210	No drop file.
212	The pointer to the system message Alpha does not exist.
213	The pointer to the system message Alpha was out of bounds.
215	No error exit address was specified, and the system message encountered an error.
Bxx	File already extended to maximum.
Cxx	Attempt to read past EOF on a file.
Dxx	File segment space in FILEI is used up.
Exx	No disk space is available for extension of a file.

Table B-4. Tape Error Codes (Sheet 1 of 5)

Code	Meaning
I/O Errors	
001	Call not in user range.
002	Illegal subfunction code (sfnc).
003	Nonexistent input/output connector (IOC).
004	Buffer size greater than 48 pages.
005	Tried to write zero length logical tape record (V tape format).
007	PRU read is longer than MPRU. Device capacity exceeded.
008	LRU is greater than MPRU.
009	WRITE attempted a zero-length PRU.
010	User WRITE buffer went minus.
011	HDR1 label not in label buffer.
012	Non-numeric file sequence number.
013	Section 1 is not in VSN list.
014	Cannot swap backwards, no previous VSN.
016	File accessibility characters do not match.
017	Position not found in multifile set.
019	Next VSN was not given.
020	Tape file does not have proper access.
021	Read or skip forward after write (illogical sequence).
030	Attempt to reuse call before previous call is complete.
031	Previous call for this unit had a fatal error.
032	Call crosses page boundary.
033	IOC is not for a tape file.
034	Tape is not assigned to this user.
037	For write operation, sum of LRU sizes is greater than buffer length.
039	Forward motion attempted when end-of-information has been detected on this file.

Table B-4. Tape Error Codes (Sheet 2 of 5)

Code	Meaning
I/O Errors	
040	End-of-tape encountered. (This number is returned to the user only if the user selected end-of-tape processing on the Q5OPEN call.)
041	Load point encountered on tape from backward motion.
042	Tape format mismatch.
043	EOI encountered while positioning to HDRI.
044	Illegal user labels in label buffer.
045	Small and large pages exist in the buffer.
046	System tables full, try again.
047	I/O request currently outstanding for this buffer.
048	Length of LRU array less than 1 or greater than 255.
049	Attempted to write over unexpired label.
050	Buffer size smaller than MPRU for read data function.
051	Tried to write two consecutive tape marks.
052	Data in LRU array after end-of-group.
053	Buffer length is less than MPRU.
054	Tape unit not assigned to any user.
055	All hardware paths to tape unit are down.
101	Tape that should be labeled is unlabeled.
102	Tape that should be unlabeled is labeled.
105	Write parity error unrecoverable.
106	Unrecognizable label group.
107	Header label fields do not match.
108	Record fragment encountered.
109	ATS software error.
110	Unexpected load point detected.
111	Read parity error unrecoverable.

Table B-4. Tape Error Codes (Sheet 3 of 5)

Code	Meaning
I/O Errors	
112	Unrecognizable trailer label.
113	Cannot read label group.
115	ATS hardware error, see hardware status.
116	Position uncertain, ready dropped.
117	Unrecoverable erase parity error.
118	Unrecoverable tape mark parity error.
119	Tape positioning parity error.
120	Unit reserved by other controller.
122	Tape mark write verify failure.
123	Blank tape encountered during read.
125	Tape repositioning error; block ID mismatch.
126	Tape repositioning error; invalid block ID.
128	Channel malfunction I/O suspended by driver.
129	Multifile position uncertain.
131	Device ID burst fault. Remount on any unit.
132	Device tape cleaner fault. Remount on any unit.
135	Tape unit switched offline.
136	No write enable ring in reel.
137	Controller not capable of requested density.
138	Unexpected error returned by ATS controller.
139	Software interface error between NADS.
140	TAD hardware error.
141	Write verify error.
142	Unit remained busy after rewind.
143	Unit dropped ready during rewind.
145	Illegal user level number.

Table B-4. Tape Error Codes (Sheet 4 of 5)

Code	Meaning
I/O Errors	
146	Label reposition error.
147	Unit reset status active, position uncertain.
148	Tape label not multiple of 80 characters.
149	Unit is not ready at reserve time.
150	No file mark after EOF1.
151	Missed file mark.
152	No label block after file marks.
153	VOL1 not detected after load point.
160	Encountered two tape marks in reverse.
170	No current block count given for find position.
171	Latest block ID not file mark or load point.
172	No label found on labeled tape.
173	Cannot find position in find position.
174	Tape mark encountered position found.
175	Compare count over block ID count.
Network Hardware Errors	
208	Channel not reserved.
209	Channel active from system.
210	No inactive to function.
211	No inactive to status function.
213	No status returned.
214	Incomplete status returned.
215	Function parameters not accepted.
216	All data not accepted from DI (device interface).
217	Incomplete data returned to DI (device interface).

Table B-4. Tape Error Codes (Sheet 5 of 5)

Code	Meaning
Network Hardware Errors	
218	Unit not connected.
221	Abnormal general status.
222	Subsystem hung.
223	Unit hung busy.
224	Interface dead.
225	Buffer index or WC error.
227	Data compare error - MALET.
228	No end of operation.
230	Command sequence error.
231	Parameter out of legal range.

GLOSSARY

C

The following terms are used in this manual. Refer to the ANSI Vocabulary for Information Processing Standard for terms not included in this glossary.

Abnormal Termination

The procedure the system follows when a batch task returns a completion code greater than the error threshold value for the batch job.

Access

Permitted use of a file or files. For example, a user with access to a pool can use the files that belong to the pool. A user with one or more access permissions to a file can use the file in the permitted modes.

Account Block

The number of system resources accumulated per charge number.

Account Identifier

One through eight characters indicating who is to be charged for system resources used by a job.

Account Number

One to eight characters indicating who is to be charged for system resources used by a job.

Allocation Unit

An amount of disk space, in 512-word blocks which is used by the system as a guide to allocating a file when a write occurs beyond the extent of the current file space.

Autoload

The process of starting operating system execution (also known as system deadstart).

Batch Deck

A card deck that begins with a STORE card and that ends with a card having the digits 6, 7, 8, and 9 multipunched in column 1.

Batch Input File

A mass storage file containing the control statements, programs, data, and directives that define a batch job.

Batch Job

A sequence of tasks the batch processor executes for a user number. The tasks are requested by batch execute lines in a batch input file.

BATCHPRO

Refer to Batch Processor.

Batch Processor

A system utility that processes batch jobs. Each batch task is executed as a controllee of the batch processor.

Beginning of Information (BOI)

The point in a file before which no data exists.

For labeled tape files, the BOI is the point immediately after the HDR1 label; for unlabeled tape files, the BOI is the load point of the tape volume.

Block

The smallest quantity of data that can be read or written by one device access. On CYBER 200 mass storage, a block is 512 64-bit words.

BOI

Refer to Beginning of Information.

Byte

A sequence of eight bits that is a subdivision of a word and is sufficient to represent a single character.

Charge Number

Combination of the account identifier and project number that is to be charged for system resources used by a job or interactive session.

Checkpoint

A user checkpoint is a system feature that captures a task and any of its controllees at some point into execution such that the task can be restarted from that point. Checkpoint can be called through a FORTRAN program named CHKPNT.

A system checkpoint is a procedure that captures the system state for later analysis. It saves the information necessary to restart the system where it was interrupted.

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. The user can usually place a comment after the command terminator.

Controllee

A task started by another task (its controller).

Controllee Chain

An ordered set of tasks. Except for the first and last tasks in the chain, each task was started by the task at the next lower level (its controller) and starts the task at the next higher level (its controllee). The chain can comprise up to nine tasks.

Controllee File

An executable file generated by the LOAD utility.

Controller

A task that starts another task (its controllee).

CPU

Central processing unit; the computational facility of the CYBER 200 system.

Data Base

A special common block created by the compiler that contains the preset data for the register file when a subroutine is entered.

Data File

A nonexecutable file, also called a physical file or physical data file.

DAU

Refer to Device Allocation Unit.

Dayfile

A file in which VSOS maintains a history of processing events. See also Job Dayfile and System Dayfile.

Default Project Number

A project number that can be assigned to a user number as default. Whenever a user executes a job or interactive session, the system resources accumulated will be charged to the default project number, if in existence, unless the user supplies a charge number within the job or interactive session and, if the user specifies an account identifier on the user statement or LOGIN statement.

Device Allocation Unit (DAU)

The basic unit of disk space management for a device; all allocation units for file segments on a device are multiples of the DAU.

Device Number

The ordinal of a disk or tape drive in the System Configuration Table.

Device Overflow

The system feature which allows segments of a single file to be resident on more than one device; allocation on a new device is usually based on the previous one being full.

Device Set

A logical grouping of physical disk devices which the system treats as a single area of device space or file residence; a device set defines the limit on device overflow; a file resides within one device set and may not overflow to another.

Directive

Supplementary control information in a file required in addition to a utility call. Directives are required, for example, with UPDATE.

Drop File

A file the system creates for each task to which the system maps modified pages from the task's virtual space. Drop file names are formed by the system by shifting the controllee name right one character, truncating the rightmost character, and adding one digit as the leftmost character of the name. The digit added corresponds to the controller level of the task.

DVSTnn

The device set name where nn is the device set number.

Dynamic Allocation

The assignment of additional space incrementally to a file as the file is being written.

Dynamic Library

A library of modules that can be dynamically loaded and linked.

Dynamic Linking

An execution-time process that locates an external subroutine and causes a transfer of control to the subroutine.

Dynamic Loading

An execution-time process that locates an external subroutine, initializes its common blocks and data base, and causes transfer of control to the subroutine. The subroutine code is not moved.

Dynamic Module

A module that contains unprocessed loader text. The text does not modify the code.

End of File (EOF)

The point in a file after which no data can be accessed. The R, W, and L record formats mark the EOF with a file delimiter.

End of Information (EOI)

The point in a file after which no data exists.

For labeled tape files and non-V format unlabeled tape files, the EOI is the point before the EOF1 label. For V format unlabeled tape files, the EOI is indicated by two consecutive tape marks.

EOF

Refer to End of File.

EOI

Refer to End of Information.

Explicit Input/Output

A means of accessing a file in which data is buffered under program control.

Family

A set of print files generated by the tasks in a batch job.

File

A collection of data that can be accessed by file name. Unless otherwise indicated, all references to files in this manual assume mass storage files.

FILEI

Refer to File Index Table.

File Index Table (FILEI)

This term refers to the file index table or to an entry in this table. The file index table is a catalog of files. Each catalog entry describes the file. Output from the AUDIT and FILES utilities shows much of the table information.

File Information Table (FIT)

A table generated for each file so that SIL can coordinate I/O processing for the file.

File Type

A mass storage file characteristic that determines if the file is executable. The file types are controllee (virtual code) file and physical data file.

FIT

Refer to File Information Table.

Full-Length PRU

A tape PRU that is the MPRU size. Contrast with Short PRU.

Group

A set of data within a file consisting of one or more records. Groups can exist within R and W format files.

IBG

Refer to Interblock Gap.

Implicit Input/Output

A means of accessing a mass storage file in which the system brings a page of the file into central memory in response to a reference to that page.

Input/Output Connector (IOC)

An entry in a minus page that links a mass storage file with a task.

Input Queue Manager (IQM)

The system routine that determines when a batch job is given to the CPU scheduler. IQM processes the RESOURCE control statement.

Interactive Session

A sequence of tasks keyed from a terminal. The duration of an interactive session is from login to logout without a task running.

Interblock Gap (IBG)

Space between tape blocks. The space is required to start the tape moving at the appropriate speed for reading or writing data.

Invisible Package

A hardware convention that contains the address and control information for the corresponding task.

IOC

Refer to Input/Output Connector.

IQM

Refer to Input Queue Manager.

JDO

Refer to Job Descriptor Ordinal.

JDN

Refer to Job Descriptor Number.

Job

A sequence of tasks related by a specific set of characteristics and initiated from a single point (for example, batch commands in one job stream or interactive in one session). Each job has a job descriptor table (JDT) entry that describes the job characteristics.

Job Block

The number of system resources accumulated for the duration of the job.

Job Dayfile

A file on which the history of a job is recorded. The job dayfile is printed at the end of the job output.

Job Descriptor Ordinal (JDO)

The number of a physical entry in the job descriptor table for a specific job.

Job Descriptor Number (JDN)

A unique number (1 through 2047) assigned to a job for the life of the job in the CYBER 200 (whether in the input, execute, or output queue, or checkpointed).

Job Descriptor Table

A system table containing information specific to a job. This includes job/session name, job step or task name, priority, working set, number of tapes committed, number of tapes assigned, pointers to descriptor blocks, and pointer to user activity table.

Large Page

128 512-word blocks; 65536 contiguous 64-bit words (contrast with Small Page).

Last-Group-File

Identifies the member of an output-file-family which contains disposition information for QTF. Its name can have one of two forms:

PYYxxxxxx
Q5Lxxxxxx

Where:

xxxxxx Five-character hash code derived from the system clock.

PYY Indicates that the output-file-family is the output of a batch job.

Q5L Indicates that the output-file-family is the result of MFQUEUE.

Level Number

A character in the range 0 through 9 or A through F written in the LRU terminator. Level number use may be required for tape interchange with a CYBER 170 system.

Library

A file of modules generated by the OLE utility that the LOAD utility can use to satisfy external references during generation of a controllee file.

LID

Refer to Logical Identifier.

Local File

A private file that is destroyed by the system after termination of the batch job or terminal session that created the file.

Logical Identifier (LID)

Nonunique identifier for a mainframe for the purpose of identifying its logical function in the LCN.

Logical Record Unit (LRU)

One or more tape PRUs read or written by a single I/O operation. For V tape format, an LRU is equivalent to a PRU. For I, SI, and LB formats, an LRU is one or more PRUs ending with a short PRU containing a terminator.

Loosley Coupled Network (LCN)

The high-speed communications network used by CYBER 200 to transfer permanent files and batch jobs and to support interactive access.

LRU

Refer to Logical Record Unit.

Map

1. Process of assigning a physical address range to a virtual address range.
2. Table containing the correspondence between virtual address ranges and physical address ranges.

Mass Storage

1. In general, mass storage is disk storage.
2. Specifically, a file management category that indicates no special file processing after task termination.

Master Project Number

One to three characters to be assigned to a mass storage file. These characters are the first three nonspecial characters of a project number.

Master User

A user who has been designated to be in charge of an account number and/or charge number and who is allowed to view, alter and/or create user descriptions within their specific accounts.

Minus Page

The first page of a virtual file used by the system to hold items such as the invisible package, input/output connector information, and maps of defined virtual space. Drop files contain a second minus page.

Multifile Set

A set of tape files that can be accessed by a single tape file request. The files are contiguous; each is delimited by its HDR1 and EOF1 labels. The multifile set can extend for one or more tape volumes.

Nonprivileged User

A user with status that allows access to files owned by the same user number under which the task is running, to public files, and to authorized pool and private files.

Object Code File

A file generated by a compiler or assembler containing relocatable code modules.

Output File

A file destined for print or punch equipment.

Also, a generic term for a file being written, as opposed to an input file being read.

Output-File-Family

A set of files residing in the output queue that was generated as the output of a batch job or as the output of MFQUEUE. If the output-file-family is from a batch job, it has at least two members, PXXxxxxx and PYYxxxxx; where xxxxx is a five-character hash code, PXX is the dayfile of the batch job, and PYY is the last-group-file of the output-file-family. Other members of the output-file-family, if present, result from execution of job tasks within the job and have the form P00xxxxx, P01xxxxx, P02xxxxx, and so forth, corresponding to the order in which the job tasks that produced output were performed. If the output-file-family is from MFQUEUE, it has exactly two members:

Q50xxxxx The MFQUEUEd file

Q5Lxxxxx The last-group-file

Ownership

A file characteristic that determines what nonprivileged tasks can access a mass storage file. Ownership categories are private, pool, and public. Private includes local and permanent files.

Pack File Index (PFI)

A table of 16-word file index table entries that exists on each pack to control the files located on each of those packs.

Pack Number

A hexadecimal two-digit number which identifies the pack medium. It is in the last two characters of the pack name (PACKnn).

Page

The unit by which central memory is allocated (See also Large Page and Small Page).

Page Fault

Reference by virtual address to a page not currently in central memory, causing a task interrupt and paging in.

Partition

A logical delimiter of a record, group, or file.

Permanent File

A file that continues to exist after termination of the batch job or interactive session that creates it.

Permanent File Transfer Facility (PTF)

The utility used by RHF to send a request to a remote system for a permanent file transfer.

Permanent File Transfer Facility Servicer (PTFS)

The utility used by RHF to accept and process a request from a remote system for a permanent file transfer.

PFI

Refer to Pack File Index.

Physical Address

Actual central memory address.

Physical Identifier (PID)

Unique identifier of a mainframe in the LCN.

PID

Refer to Physical Identifier.

Pool

A group of files accessible by more than one user, but owned by the pool, not by an individual user. Pools are attached to jobs.

Pool Boss

The user authorized to administer a pool. The pool boss can delete pool files, destroy the pool, and grant and remove user access to the pool.

Pool File

An ownership category that indicates a file can be accessed by any privileged task and by the pool boss or any pool member.

Pool Member

A user granted access to a pool by the pool boss.

Preallocation

The assignment of space to a file before data is written.

Private File

An ownership category; a private file is owned by a user number.

Privileged User

A user with status that allows access to all permanent files in the system and to almost all operating system functions.

Procedure File

A file containing a sequence of control statements headed by a PROC statement. When the batch processor processes a BEGIN statement that specifies the procedure file, it inserts the control statement sequence from the procedure file into the control statement sequence for the job.

Production Controllee

A controllee that has been validated for production status by the site security administrator.

Production User

A user whose user number has been validated for production status by the site security administrator.

Production User Number

A user number that has been validated for production status by the site security administrator.

Project Number

One to 20 alphanumeric characters (including the special characters * and -) indicating the project within the account identifier to which to charge the system resources.

PRU

For tape files, the amount of data read or written by a single tape function (also known as a tape block). For I, SI, and LB formats, the PRU size is fixed by the system. For V format, the user can specify the PRU size.

PTF

Refer to Permanent File Transfer Facility.

PTFS

Refer to Permanent File Transfer Facility Servicer.

Public File

An ownership category that indicates a file can be accessed by all users.

Record

The smallest logical set of data defined within an SIL file format.

Remote Host

A mainframe within the LCN with which the local host can communicate.

Remote Host Facility (RHF)

The set of network applications and the supporting operating system software that address the LCN for CYBER 200, and the remote systems with which it communicates.

RMS

See Rotating Mass Storage.

Remote Workstation Facility (RWF)†

The set of network applications and the supporting operating system software that allows direct communication between the CYBER 200 and a network of workstations.

Remote Workstation Interface (RWI)†

The set of network applications and the supporting operating system software that allows direct communication between the CYBER 200 and a network of workstations.

Rotating Mass Storage

An 819 disk.

SBU

Refer to System Billing Unit.

Scalar

A data item representing a single value that is processed by a scalar machine instruction. (Refer to Vector.)

Scratch File

A file that is destroyed upon termination of the task that creates it.

Security level

Attribute of a file, task, job, or user number used to prevent unauthorized data access. The eight security levels are numbered 1 through 8, from least to greatest security.

Segment

An area of contiguous disk space allocated to a file.

Shared SYSLIB

An OLE library that was altered by SLGEN so that it can exist as a dynamic library in the system shared library file. It contains all the modules on a normal SYSLIB.

Shared Utility

A static module set that consists of loaded code that resides in the system shared library file and a controllee file containing everything but the code.

Short PRU

A tape PRU of less than the MPRU size. A short PRU is the last PRU in a logical record unit (LRU) for I, SI, and LB formats. (V format does not support short PRUs.) The LRU terminator is appended to the short PRU. Refer also to Zero-length PRU. Contrast with Full-length PRU.

SHRLIB

Refer to System Shared Library.

†The necessary software and hardware to support workstation connections. This is a product of ETA Systems Inc.

SIL

Refer to System Interface Language.

Site Security Administrator

The individual(s) whose responsibility it is to maintain and grant production status to users and controllees.

Small Page

The smaller of the two page sizes (contrast with Large Page). The small page size is chosen during VSOS autoloading; the possible sizes are one, four, or sixteen 512-word blocks.

Source File

1. A generic term for a file containing text read by a utility or other task.
2. In an UPDATE utility context, a file produced by UPDATE that would allow recreation of a new program library on a subsequent creation run.

STU

Refer to System Time Unit.

System Billing Unit (SBU)

An installation-defined unit used for charging system resources. The unit might incorporate tape use/access, number of tape functions, number of disk accesses, number of pages transferred to or from disk, and CPU usage in microseconds. An example of SBU is time in microseconds of CPU use.

System Dayfile

A file on which VSOS records information on all tasks. A privileged user can send a message to the system dayfile.

System Interface Language

A set of subroutines that user programs can call to perform system functions.

System Message

The means by which the operating system and user tasks communicate with each other. System messages, which are formatted in Alpha words and Beta words, are described in volume 2 of this reference manual.

System Pool

A pool of files used instead of the public files with the same names. If a system pool exists in the current system, the pool is attached to each user number when the user number submits a job or logs in.

System Shared Library (SHRLIB)

A file that is read into shared pages at system initialization. It includes directories, shared utilities, and a shared SYSLIB.

System Time Unit (STU)

An installation-defined unit used for allocating system resources. The unit might incorporate number of disk accesses, number of pages transferred to or from disk, and CPU usage in microseconds. An example of STU is time in microseconds of CPU use.

System User

A user executing on the system user number.

Task

One execution of a program; job step; controllee. Task characteristics are defined by a Descriptor Block (DB).

Threshold Value

The maximum return code that a task can return without causing the batch processor to initiate abnormal job termination. The user sets the threshold value with the TV control statement.

UPC

Refer to User Project Control.

User Number

A 6-digit identifier that specifies a file owner or user of system resources.

User Project Control

An attribute which can be set for a user number so that the charge number must be specified for the executing job or the user must have a default project number assigned.

Vector

A set of data items specified as a single operand for a vector machine instruction. Execution of the vector instruction processes each data item in the set.

Virtual Address

Address referring to virtual memory and translated, by the page table, into a physical address.

Virtual Code File

Refer to Controllee File.

Virtual Memory

A means of addressing memory in which the system maps the addresses referenced by a task to actual physical addresses in memory.

Volume

A reel of magnetic tape.

Word

A division of central memory or mass storage corresponding to 64 bits. Bits are numbered 0 through 63 left to right.

Working Set

The pages of a task's virtual space that are most frequently referenced during task execution. The size of its working set determines when the task can be scheduled for CPU use.

Zero-length PRU

A short tape PRU containing no data. The PRU consists solely of the LRU terminator. Refer to Short PRU.

SIL FILE INFORMATION TABLE

D

SIL coordinates I/O processing for a file, using the file information table (FIT) generated for the file. Table D-1 shows the structure of the FIT. Figure D-1 lists, in more detail, the contents of the FIT.

Table D-1. File Information Table

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
peof	lfn																																																															64
conv	st	bt	rt	lfp	of	cf	oc	os	flags	reserved	pc	rmk																																																			8	
				llop				lvsc				lvso				reserved												16																																				
	res	buf1			buf1				buf1												48																																											
	res	buf2			buf2				buf2												48																																											
	acs	try	wsa										48																																																			
	wsl																reserved																32																															
	ioer				rsc				ctfp				bnw								24																																											
	rc																bn																32																															
	mnr																mxr																32																															
	rl																ptl																32																															
	res	ioc				cbo												48																																														
	lt	ado	den	gvn	rp	dt	scn				fsn				16																																																	
	rsn	cltyp	unit	unused	gs	fs				20																																																						
	fid1																																																															64
	fid2																																																															64
	fid3	stid										acst												8																																								
	gn																vsn																48																															
	lvsn																vsna																48																															
bfig	scad				len												44																																															
	l	e	w	l	res	rfd	cerr	aord	serr	fadd				12																																																		
	reserved																tbl																32																															
	sbn1												eod1												40																																							
	sbn2												eod2												40																																							
	ownr																																																															64
	ownr2																																																au				16											
	nbyte																hbyte																32																															
	rpb	lp	dn	ns	tf	cm	brsn	unused																																																								8
38-53	filei copy																																																															64
	tbn																mpru																32																															
	mfn																																																															64
	dtt																																																															64
	brl																bstat																32																															
	reserved																												ibo				iwo												24																			
59-63	unused																																																															64

<u>Word</u>	<u>Bits</u>	<u>Field</u>	<u>Contents</u>
0	0-63	lfn	File name; eight ASCII characters.
1	0	sfo	SIL file organization: 0 Sequential access. 1 Direct access.
1	1-3	bt	Blocking type: 0 Non-SIL file. 1 Internal blocking (I). 2 Character count blocking (C). 4 Record count blocking (K).
1	4-7	rt	Record type: 0 Control word (W). 1 ANSI fixed length (F). 2 Record mark delimited (R). 4 CYBER Record Manager control word (L). 5 System block (B). 7 Undefined (U).
1	8	peof	Physical end-of-file flag (end of information).
1	9-15	lfp	Logical file position: 0 Within a logical record. 1 Beginning of information (BOI). 2 Beginning of a logical file (BOF). 3 Within a file (between its BOF and EOF). 4 End of volume (EOV) 5 End of logical file (EOF). 8 End of logical group (EOG). 16 End of logical record (EOR). 32 Beginning of volume (BOV) 64 End of information (EOI).
1	16-17	ofp	File positioning when opened: 0 Do not rewind the file. 1 Rewind the file. 2 Position the tape file according to the block ID and PRU counts in the system tapes table.
1	18-19	cfp	File positioning when closed: 0 No rewind. 1 Rewind. 2 Rewind and unload.

Figure D-1. FIT Format (Sheet 1 of 7)

<u>Word</u>	<u>Bits</u>	<u>Field</u>	<u>Contents</u>
1	20-21	conv	Tape data conversion mode: 0 Binary data; do not convert. 1 Character data; convert to and from character codes.
1	22-23	ocs	Open or closed file status: 0 Never opened. 1 Opened for explicit I/O. 2 Closed. 3 Opened for implicit I/O.
1	24	srf	Flag indicating whether control is returned immediately to the caller after issuance of an I/O request; if set, control is not returned until the I/O request is complete.
1	25	wpf	Write flag; if set, the last operation was a write operation.
1	26	bsf	Buffer specified flag; if set, Q5OPEN, Q5GETFIT, or Q5SETFIT specified a buffer.
1	27	pef	Parity error flag set if parity violation encountered RT=W.
1	28	cef	Compression/expansion flag; if set, blank compression and expansion is performed on the file.
1	29	ppf	Put flag; if set, a put operation was the last operation performed on the file.
1	30	riof	Random I/O flag; if set, a Q5SETFIT call has changed the block numbers. The flag is used when a FORTRAN read request specifies a file address.
1	31	al	Flag indicating whether the file is accessed at the record level or the block level (used internally by access permission checking).
1	32	sra	Stop read ahead flag (for SIL internal use only).
1	33	uep	User error processing flag. Indicates whether the user gets control after VSOS returns a tape I/O error. 0 User does not get control. 1 User gets control.
1	34	ring	Ring specification.
1	35	mfs	Multifile set indicator: 0 File does not belong to a multifile set. 1 File belongs to a multifile set.
1	36-47		Reserved.

Figure D-1. FIT Format (Sheet 2 of 7)

<u>Word</u>	<u>Bits</u>	<u>Field</u>	<u>Contents</u>																																																								
1	48-55	pc	Padding character; one ASCII character.																																																								
1	56-63	rmk	Record mark character; one ASCII character.																																																								
2	0-15	llop	Last logical operation on the file requested by the user: <table style="margin-left: 40px;"> <tr><td>1</td><td>Q5DEFINE</td><td>15</td><td>Q5READ</td></tr> <tr><td>2</td><td>Q5MAPIN</td><td>16</td><td>Q5WRITE</td></tr> <tr><td>3</td><td>Q5MAPOUT</td><td>17</td><td>Q5REWIND</td></tr> <tr><td>4</td><td>Q5OPEN</td><td>18</td><td>Q5SKIP</td></tr> <tr><td>5</td><td>Q5CLOSE</td><td>19</td><td>Q5CHANGE</td></tr> <tr><td>6</td><td>Q5PURGE</td><td>20</td><td>Q5GIVE</td></tr> <tr><td>7</td><td>Q5RETURN</td><td>21</td><td>Q5REDUCE</td></tr> <tr><td>8</td><td>Q5REQUEST</td><td>22</td><td>Q5ROUTE</td></tr> <tr><td>9</td><td>Q5CHECK</td><td>23</td><td>Q5LABEL</td></tr> <tr><td>10</td><td>Q5ENDPAR</td><td>24</td><td>Q5REELSW</td></tr> <tr><td>11</td><td>Q5GETN</td><td>25</td><td>Q5CLIOER</td></tr> <tr><td>12</td><td>Q5GETP</td><td>26</td><td>Q5PERMIT</td></tr> <tr><td>13</td><td>Q5PUTN</td><td>27</td><td>Q5GETB</td></tr> <tr><td>14</td><td>Q5PUTP</td><td>28</td><td>Q5PUTB</td></tr> </table>	1	Q5DEFINE	15	Q5READ	2	Q5MAPIN	16	Q5WRITE	3	Q5MAPOUT	17	Q5REWIND	4	Q5OPEN	18	Q5SKIP	5	Q5CLOSE	19	Q5CHANGE	6	Q5PURGE	20	Q5GIVE	7	Q5RETURN	21	Q5REDUCE	8	Q5REQUEST	22	Q5ROUTE	9	Q5CHECK	23	Q5LABEL	10	Q5ENDPAR	24	Q5REELSW	11	Q5GETN	25	Q5CLIOER	12	Q5GETP	26	Q5PERMIT	13	Q5PUTN	27	Q5GETB	14	Q5PUTP	28	Q5PUTB
1	Q5DEFINE	15	Q5READ																																																								
2	Q5MAPIN	16	Q5WRITE																																																								
3	Q5MAPOUT	17	Q5REWIND																																																								
4	Q5OPEN	18	Q5SKIP																																																								
5	Q5CLOSE	19	Q5CHANGE																																																								
6	Q5PURGE	20	Q5GIVE																																																								
7	Q5RETURN	21	Q5REDUCE																																																								
8	Q5REQUEST	22	Q5ROUTE																																																								
9	Q5CHECK	23	Q5LABEL																																																								
10	Q5ENDPAR	24	Q5REELSW																																																								
11	Q5GETN	25	Q5CLIOER																																																								
12	Q5GETP	26	Q5PERMIT																																																								
13	Q5PUTN	27	Q5GETB																																																								
14	Q5PUTP	28	Q5PUTB																																																								
2	16-31	lvsc	Last virtual system function code issued for the file.																																																								
2	32-47	lvso	Last virtual system suboperation code issued for the file.																																																								
2	48-63		Reserved.																																																								
3	0-2		Reserved.																																																								
3	3-15	buf11	Length in 512-word blocks of buffer 1.																																																								
3	16-63	buf1	Address of buffer one.																																																								
4	0-2		Reserved.																																																								
4	3-15	buf12	Length in 512-word blocks of buffer 2.																																																								
4	16-63	buf2	Address of buffer 2.																																																								
5	0-7	acs	File access permissions (each bit is a flag for an access permission; any combination of permissions is valid). <table style="margin-left: 40px;"> <tr><td>1</td><td>Write</td></tr> <tr><td>2</td><td>Read</td></tr> <tr><td>4</td><td>Append</td></tr> <tr><td>8</td><td>Modify</td></tr> <tr><td>16</td><td>Execute</td></tr> </table>	1	Write	2	Read	4	Append	8	Modify	16	Execute																																														
1	Write																																																										
2	Read																																																										
4	Append																																																										
8	Modify																																																										
16	Execute																																																										
5	8-15	try	Error recovery for tape files: <table style="margin-left: 40px;"> <tr><td>0</td><td>Standard error recovery procedures.</td></tr> <tr><td>1</td><td>No attempt to recover errors.</td></tr> </table>	0	Standard error recovery procedures.	1	No attempt to recover errors.																																																				
0	Standard error recovery procedures.																																																										
1	No attempt to recover errors.																																																										

Figure D-1. FIT Format (Sheet 3 of 7)

<u>Word</u>	<u>Bits</u>	<u>Field</u>	<u>Contents</u>
5	16-63	wsa	Address of the working storage area.
6	0-31	wst	Length in bytes of the working storage area.
6	32-63		Reserved.
7	0-15	ioer	Tape I/O error returned by VSOS.
7	16-31	rsc	Residual tape partition skip count.
7	32-39	ctfp	Current tape position as returned by VSOS.
7	40-63	bcw	Number of bytes to next control word (for SIL internal use only).
8	0-31	rc	Record count; number of the last full record read or written.
8	32-63	bn	Ordinal of current block.
9	00-31	mnr	Minimum record length in bytes.
9	32-63	mrx	Maximum record length or fixed record length in bytes.
10	00-31	rl	Current record length in bytes.
10	32-63	ptl	Current partial transfer length.
11	0-1	cbn	Current buffer being used: 0 Both buffers free. 1 Buffer one in use. 2 Buffer two in use.
11	2-7		Reserved.
11	8-15	ioc	Number of the system I/O connector (IOC) used by the file.
11	16-63	cbo	Current byte position within the buffer.
12	0-2	lt	Tape label type: 1 ANSI standard labels. 2 No labels (unlabeled tape). 3 Nonstandard labels.
12	3-5	ado	Assembly/disassembly option: 0 Binary. 1 60/64 format.

Figure D-1. FIT Format (Sheet 4 of 7)

<u>Word</u>	<u>Bits</u>	<u>Field</u>	<u>Contents</u>
12	6-7	den	Tape recording density: 1 6250 cpi. 2 1600 cpi.
12	8-15	gvn	Tape file generation version number from HDR1 label.
12	16-27	rp	Tape file retention period; used to determine the expiration date in the HDR1 label.
12	28-31	dt	Device type: 0 Mass storage. 1 Interactive terminal. 9 Magnetic tape.
12	32-47	scn	Tape file section number from HDR1 label.
12	48-63	fsn	Tape file sequence number from HDR1 label.
13	00-04	vsn	Request serial number of last Q5READ or Q5WRITE call. It also identifies the word within the FIT (st1 through st6) containing the call response.
13	05-07	cltyp	Close type (as determined by the Q5OPEN call): 0 Nonprivileged. 1 Privileged. 2 Privileged system task.
13	08-15	unit	Logical unit number of the device on which the file resides as set by the system.
13	16-21		Reserved.
13	22	gsf	Flag indicating whether a group separator is in the current I/O buffer (only valid for R record format). 0 No group separator is in the buffer. 1 A group separator is in the buffer.
13	23-42	gs	Byte offset giving the location of the group separator in the I/O buffer currently in use (only valid for R record format).
13	43	fsf	Flag indicating whether a file separator is in the current I/O buffer (only valid for R record format). 0 No file separator is in the buffer. 1 A file separator is in the buffer.
13	44-63	fs	Byte offset giving the location of the file separator in the I/O buffer currently in use (only valid for R record format).

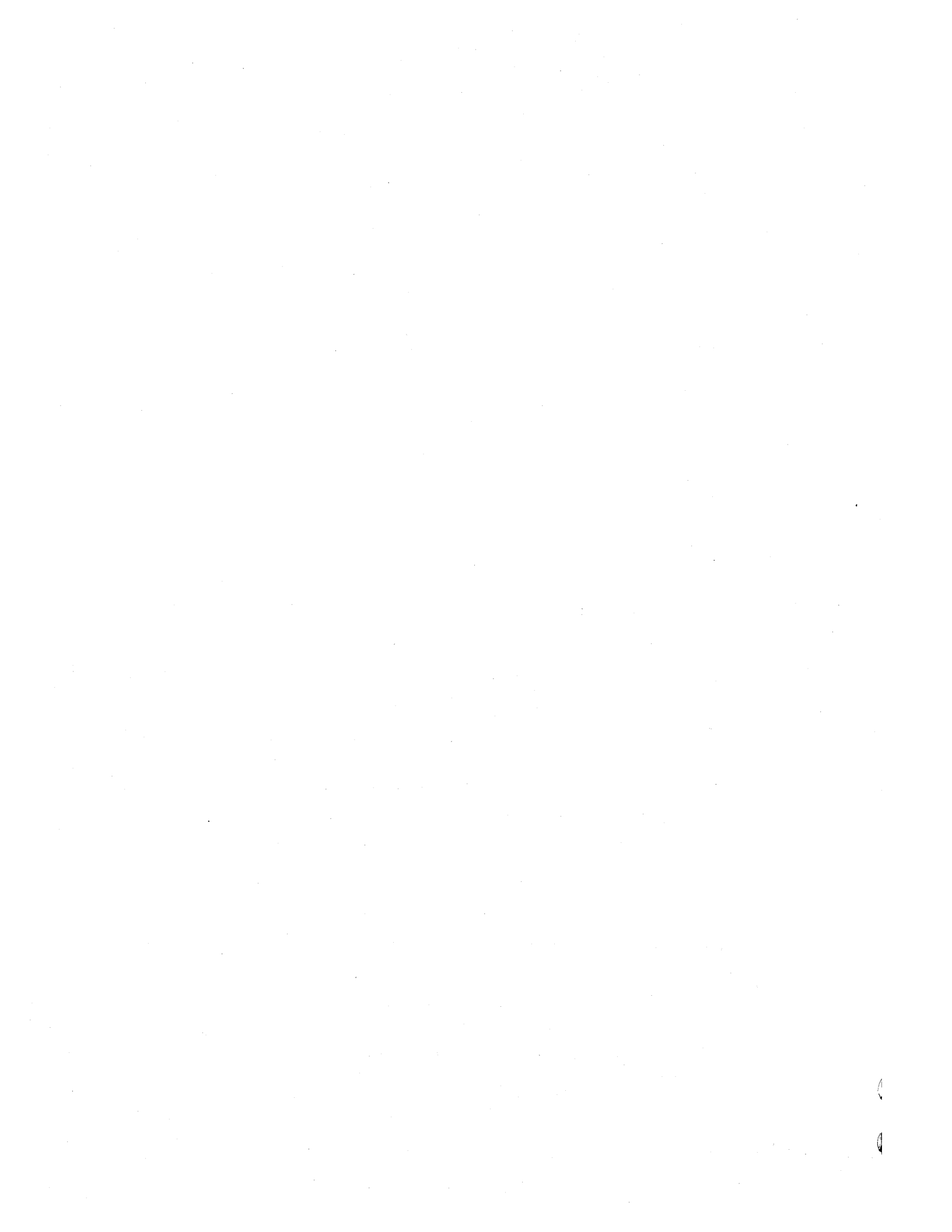
Figure D-1. FIT Format (Sheet 5 of 7)

<u>Word</u>	<u>Bits</u>	<u>Field</u>	<u>Contents</u>
14	0-63	fid1	First eight characters of tape file identifier from HDR1 label.
15	0-63	fid2	Second eight characters of tape file identifier from HDR1 label.
16	0-7	fid3	Last character of tape file identifier from HDR1 label.
16	8-55	stid	File set identifier for a tape file.
16	56-63	acst	File accessibility character for HDR1 label.
17	0-15	gn	Tape file generation number for HDR1 label.
17	16-63	vsn	Volume serial number for VOL1 label.
18	0-15	lvsn	Length in words of the user-specified VSN list.
18	16-63	vsna	Address of user-specified VSN list.
19	0-19	scad	Beginning sector address of the file.
19	20-63	len	Number of bytes of data transferred by the last I/O operation.
20-21	0-63	st1	Status words used by an internal routines for physical I/O operations.
22-23	0-63	st2	
24-25	0-63	st3	
26-27	0-63	st4	
28-29	0-63	st5	
30-31	0-63	st6	
32	0-23	sbn1	Starting block number in buffer one.
32	24-63	eod1	Byte offset to end of data in buffer one.
33	0-23	sbn2	Starting block number in buffer two.
33	24-63	eod2	Byte offset to end of data in buffer two.
34	0-63	ownr	First eight characters of owner identification in VOL1 label.
35	0-47	ownr2	Last six characters of owner identification in VOL1 label.
35	48-63	au	Allocation unit.
36	0-31	nbyte	Next available byte in the file.
36	32-63	hbyte	Highest available byte in the file.
37	0-15	rpb	Records per block for K blocked files.

Figure D-1. FIT Format (Sheet 6 of 7)

<u>Word</u>	<u>Bits</u>	<u>Field</u>	<u>Contents</u>
37	16-19	lp	Label processing option: 0 Read and verify existing labels. 1 Write new labels.
37	20-31	dn	Device number.
37	32-39	ns	Noise size in frames (0 through 31).
37	40-43	tf	Tape data format: 1 Large block (LB). 2 SCOPE internal (SI). 3 Internal (I). 4 Variable (V). #C Non-ANSI interchange - V format
37	44-47	cm	Character data conversion mode: 1 ASCII character set. 2 EBCDIC character set.
37	48-55	brsn	Request serial number (RSN) for buffer operations.
37	56-63		Reserved.
38-53	0-63	filei	Copy of file index (FILEI) entry.
54	0-31	tbn	Current block count for tapes.
54	32-63	mpru	Maximum tape block (PRU) size in bytes.
55	0-63	mfn	Multifile set name (ASCII characters; used only if the mfs field is 1).
56	0-63	dtm	File descriptor in the system tapes table.
57	0-31	brl	Record length for B type records.
57	32-63	bstat	Status of B type I/O operation (for internal use only).
58	0-27		Reserved.
58	28-39	ibo	I blocked tape file block ordinal.
58	40-63	iwo	I blocked tape file word offset.
59	0-63		Reserved.
60-63	0-63		Reserved for site use.

Figure D-1. FIT Format (Sheet 7 of 7)



FORTRAN DATA CONVERSION ROUTINES

E

As an aid to the FORTRAN programmer, the standard system library, SYSLIB, contains routines for converting IBM† and CYBER 170 arithmetic data formats to and from CYBER 200 arithmetic data formats.

IBM ARITHMETIC CONVERSION ROUTINES

The available IBM arithmetic conversion routines support the following conversions:

<u>Routine</u>	<u>Conversion</u>
Q9IC64	IBM 64-bit floating point representation to CYBER 200 64-bit floating point representation.
Q9IC32	IBM 32-bit floating point representation to CYBER 200 32-bit floating point representation.
Q9CI64	CYBER 200 64-bit floating point representation to IBM 64-bit floating point representation.
Q9CI32	CYBER 200 32-bit floating point representation to IBM 32-bit floating point representation.

The special call statement Q8EXTV converts CYBER 200 32-bit floating point representation to CYBER 200 64-bit floating point representation. The special call statement Q8CONV converts CYBER 200 64-bit floating point representation to CYBER 200 32-bit floating point representation. These routines can be used with the IBM data representation conversion routines. For example, to convert a CYBER 200 64-bit floating point number to an IBM 32-bit floating point number, use a Q8CONV statement to convert from 64-bit to 32-bit representation and then call the Q9CI32 routine. To reverse the conversion, call the Q9IC32 routine and then use the Q8EXTV statement to convert from 32-bit to 64-bit representation. For more information on the Q8EXTV and Q8CONV statements, refer to the CYBER 200 FORTRAN Language 2 Reference Manual.

For best performance, the input array for these routines should contain at least 64 operands.

IBM TO CYBER 200 64-BIT FLOATING POINT CONVERSION

The Q9IC64 routine converts numbers from IBM 370 64-bit floating point representation to CYBER 200 64-bit floating point representation.

The IBM 64-bit floating point representation has an 8-bit exponent and a 56-bit coefficient. The CYBER 200 64-bit floating point representation has a 16-bit exponent and a 48-bit representation. Therefore, when Q9IC64 converts a floating point number, six to nine of the least significant bits of the coefficient are truncated.

†IBM is a registered trademark of International Business Machines Corporation.

The data flag branch manager is disabled during data conversion and reenabled after completion of the conversion.

The following is the Q9IC64 call format:

CALL Q9IC64(arrayibm,arrayc20,num,istat)

arrayibm	Name of the array of IBM 64-bit operands to be converted.
arrayc20	Name of the array to receive the operands converted to CYBER 200 64-bit floating point representation.
num	Number of operands to convert (1 to 65 535).
istat	Name of the integer variable in which status is returned. The following are the possible return values:
0	All operands converted without errors.
-1	The number of operands specified on the call is not within the range 1 to 65 535; no operands are converted.

IBM TO CYBER 200 32-BIT FLOATING POINT CONVERSION

The Q9IC32 routine converts numbers from IBM 370 32-bit floating point representation to CYBER 200 32-bit floating point representation.

The following is the IBM floating-point representation:

number = \pm coefficient * (16.^(exponent - 64))

The following is the CYBER 200 floating point representation:

number = coefficient * 2^{exponent}

Although both 32-bit representations have 8-bit exponents and 24-bit coefficients, the IBM representation can represent larger exponents than the CYBER 200 representation. Therefore, if the exponent of an IBM operand is less than -2210, Q9IC32 converts it to CYBER 200 machine 0, and if the exponent is greater than 3310, Q9IC32 converts it to a CYBER 200 indefinite (its leftmost hexadecimal digit is 7).

Also, when the higher order bit of the coefficient is 1, the least significant bit is truncated.

The following is the Q9IC32 call format:

CALL Q9IC32(arrayibm,arrayc20,num,istat)

arrayibm	Name of the array of IBM 32-bit operands to be converted.
arrayc20	Name of the array to receive the operands converted to CYBER 200 32-bit floating point representation.
num	Number of operands to convert (1 to 65 535).

istat Name of the integer variable in which status is returned. The following are the possible return values:

- 0 All operands converted without errors.
- +n All operands converted; n operands converted to indefinites.
- 1 The number of operands specified on the call is not within the range 1 to 65 535; no operands are converted.

CYBER 200 TO IBM 64-BIT FLOATING POINT CONVERSION

The Q9CI64 routine converts numbers from CYBER 200 64-bit floating point representation to IBM 370 64-bit floating point representation.

The IBM 64-bit floating point representation cannot represent numbers whose magnitude exceeds the following value:

$$\text{coefficient} * 2^{\text{exponent}}$$

where exponent is between -306 and 205. Therefore, Q9CI64 converts numbers outside that range and indefinites to IBM zero representation where all 64 bits are zero.

The following is the Q9CI64 call format:

CALL Q9CI64(arrayc20,arrayibm,num,istat)

arrayc20 Name of the array of CYBER 200 64-bit operands to be converted.

arrayibm Name of the array to receive the operands converted to IBM 64-bit floating point representation.

num Number of operands to convert (1 to 65 535).

istat Name of the integer variable in which status is returned. The following are the possible return values:

- 0 All operands converted without errors.
- +n All operands converted; n operands converted zero due to overflow.
- 1 The number of operands specified on the call is not within the range 1 to 65 535; no operands are converted.

CYBER 200 TO IBM 32-BIT FLOATING POINT CONVERSION

The Q9CI32 routine converts numbers from CYBER 200 32-bit floating point representation to IBM 370 32-bit floating point representation.

The leftmost bit of the CYBER 200 zero representation is set to one. Q9CI32 converts CYBER 200 zeros and indefinites to IBM zero representation where all 64 bits are zero.

The following is the Q9CI32 call format:

CALL Q9CI32(arrayc20,arrayibm,num,istat)

arrayc20	Name of the array of CYBER 200 32-bit operands to be converted. The operands must be packed two per word.
arrayibm	Name of the array to receive the operands converted to IBM 32-bit floating point representation.
num	Number of operands to convert (1 to 65 535).
istat	Name of integer variable in which status is returned. The following are the possible return values:
0	All operands converted without errors.
+n	All operands converted; n indefinites converted to zero.
-1	The number of operands specified on the call is not within the range 1 to 65 535; no operands are converted.

CYBER 170 ARITHMETIC CONVERSION ROUTINES

Routines are available for the following conversions:

- CYBER 170 integer format to CYBER 200 integer format.
- CYBER 200 integer format to CYBER 170 integer format.
- CYBER 170 floating point format to CYBER 200 floating point format.
- CYBER 200 floating point format to CYBER 170 floating point format.

The routines are stored on the standard system library, SYSLIB. The routines are written in CYBER 200 FORTRAN with inline machine instruction calls to optimize performance.

CONVERSION PROCESSING

The conversion routines are intended for use within the following processing sequence:

1. A job transfers a file of arithmetic data from a CYBER 170 system to a CYBER 200 system.
2. A CYBER 200 program calls a conversion routine to convert the arithmetic data from CYBER 170 format to CYBER 200 format.
3. A CYBER 200 program uses the converted arithmetic data as operands in a calculation producing a file of result data.
4. A CYBER 200 program calls a conversion routine to convert the result data from CYBER 200 format to CYBER 170 format.
5. A job transfers the file of converted result data from the CYBER 200 system to a CYBER 170 system.

CYBER 170 arithmetic data representations use 60 bits. The conversion routines assume the data is stored as one operand in each CYBER 170 60-bit word and then transferred as a packed binary string.

The transferred CYBER 170 data is packed into the 64-bit words of the CYBER 200 file. That is, the first word of the file contains the first 60-bit operand and the top four bits of the second operand, the second word contains the bottom 56 bits of the second operand and the top eight bits of the third operand, and so forth.

The conversion routine reads the packed CYBER 170 data and converts it to one CYBER 200 operand per 64-bit word.

Similarly, the reverse conversion reads one CYBER 200 operand from each 64-bit word and produces packed CYBER 170 data. When a job transfers the packed CYBER 170 data to a CYBER 170 system, the CYBER 170 file contains one operand in each 60-bit word.

CALL FORMAT

Each conversion routine uses the same call format. The call specifies an input array containing the values to be converted and an output array to receive the converted values. To perform the conversion in place, specify the same array as the input array and the output array.

The call format for the routines is as follows:

CALL Q9xxx (array1,array2,num,istat)

Q9xxx	Name of conversion routine. The routine name is given in the conversion description.
array1	Name of the input array containing the operands to be converted. The array length must be at least the value specified as num.
array2	Name of the output array to contain the converted values. The same array can be specified as both the input array and the output array. The array length must be at least the value specified as num.
num	Number of operands to be converted. The value must be greater than zero; it can be specified as an integer constant or integer variable name.
istat	Name of an integer variable in which one of the following status values is returned: 0 No errors. -1 The specified num value was not a valid integer greater than zero. n Integer indicating the number of errors that occurred. The error type is described in the conversion description.

For optimum performance, align the input and output arrays on 512-word block boundaries. Also, for most efficient storage, the number of operands to be converted by a call should be a multiple of 8192. A packed binary string of 8192 operands fits into 15 blocks. When converted to CYBER 200 format, the 8192 operands fit into 16 blocks.

CYBER 170 TO CYBER 200 INTEGER CONVERSION

The Q9LCI routine converts CYBER 170 integer operands to equivalent CYBER 200 integer operands.

The input array must contain a packed binary string of 60-bit integers. The first operand must be left-justified in the first word of the array.

The routine converts the integers from 60-bit one's complement integers to 48-bit two's complement integers, one integer for each 64-bit word.

A value outside the range $-(2^{47}-1)$ through $2^{47}-1$ is an overflow error. Its converted value is either -2^{47} if its sign is negative or $2^{47}-1$ if its sign is positive. The number of overflow errors is returned in the istat variable.

CYBER 200 TO CYBER 170 INTEGER CONVERSION

The Q9CLI routine converts CYBER 200 integer operands to equivalent CYBER 170 integer operands.

The input array must contain 48-bit CYBER 200 integers, one integer for each 64-bit word.

The routine converts the integers from 48-bit two's complement integers to a packed binary string of 60-bit one's complement integers. The first operand is left-justified in the first word of the output array.

The routine returns the number of integers which had nonzero exponents in the istat variable.

CYBER 170 TO CYBER 200 FLOATING POINT CONVERSION

The Q9LCF routine converts CYBER 170 floating point operands to equivalent CYBER 200 floating point operands.

The input array must contain a packed binary string of 60-bit normalized floating point operands. The first operand must be left-justified in the first word of the array.

The routine converts the floating point operands to CYBER 200 64-bit normalized floating point operands. It truncates the least-significant bit of the 48-bit coefficient.

Overflow and indefinite operands are converted to CYBER 200 indefinite values. Underflow (binary zero) is converted to CYBER 200 machine zero.

Because a corresponding value exists for each CYBER 200 floating point value, no conversion errors can occur. Therefore the routine does not return a conversion error count in the istat variable.

CYBER 200 TO CYBER 170 FLOATING POINT CONVERSION

The Q9CLF routine converts CYBER 200 floating point operands to equivalent CYBER 170 floating point operands.

The input array must contain CYBER 200 64-bit normalized floating point operands.

The routine converts the floating point operands to a packed binary string of CYBER 170 60-bit normalized floating point operands. The first operand is left-justified in the first word of the output array.

Indefinite operands are converted to CYBER 170 indefinite values. CYBER 200 machine zero is converted to CYBER 170 underflow (binary zero).

If the value cannot be represented as a 60-bit floating point value (overflow error), it is converted to either positive or negative overflow depending on the sign of the overflow coefficient. The number of overflow errors is returned in the istat variable.

CYBER 170 TO CYBER 200 NUMERIC DATA TRANSFER EXAMPLE

This section outlines the steps required to transfer a file of binary data from a CYBER 170 system to a CYBER 200 system and back again. The binary data can be in floating point format or integer format.

CYBER 170 to CYBER 200 Transfer

To transfer numeric data from a CYBER 170 system to a CYBER 200 system, perform the following steps:

1. Execute the CYBER 170 program that writes the data file. The program should use a BUFFER OUT statement to write each data record. (An unformatted WRITE statement could be used instead if the job includes a FILE control statement that specifies BT=C,RT=S.)
2. Execute an MFLINK statement within a CYBER 200 job to transfer the CYBER 170 file. Specify DD=US on the statement to transfer a binary stream that preserves the logical record structure. For example, the following statement transfers a file named C170B from a NOS/BE system:

```
MFLINK,C170B,ST=XXX,DD=US,JCS="ATTACH,C170B,ID=XX."
```

3. Execute the CYBER 200 program that uses the data. The program must first read the transferred file into an array and then convert the data to the appropriate CYBER 200 data format.

The program must use BUFFER IN statements to read the file into an array. (It can use unformatted READ statements if the array bit length is a multiple of both 60 and 64. It cannot use Q7BUFIN.)

The record length used, LEN2, should be computed from the CYBER 170 record length, LEN1, using the following statements:

```
IREM=MOD((LEN1*60),64)
LEN2=(LEN1*60)/64
IF (IREM.NE.0) LEN2=LEN2+1
```

IREM is the number of bits of data in the last 64-bit word. LEN2 is the number of 64-bit words required to hold the bit string received through MFLINK.

The program calls the appropriate conversion routine to convert the data in the array. For example, the following statement converts data from CYBER 170 floating point format to CYBER 200 floating point format:

```
CALL Q9LCF(A,B,LEN1,ISTAT)
```

After executing the statement, the data in array B is available for processing by the program.

CYBER 200 to CYBER 170 Transfer

To transfer numeric data from a CYBER 200 system to a CYBER 170 system, perform the following steps:

1. Execute a CYBER 200 program that converts the data to the appropriate CYBER 170 data format and then writes it to a file.

For example, assuming the number of floating point operands in array B is LEN1, the following statement converts the data to CYBER 170 floating point format:

```
CALL Q9CLF(B,C,LEN1,ISTAT)
```

The program should then use a BUFFER OUT statement to write array C as a record on the file.

The record length specified on the BUFFER OUT statement, LEN2, should be computed from LEN1 using the following statements:

```
IREM=MOD((LEN1*60),64)
LEN2=(LEN1*60)/64
IF (IREM.NE.0) LEN2=LEN2+1
```

IREM is the number of bits of data in the last 64-bit word. LEN2 is the number of 64-bit words required to hold the bit string received through MFLINK.

2. Execute an MFLINK statement within a CYBER 200 job to transfer the CYBER 200 file. Specify DD=US on the statement to transfer a binary stream that preserves the logical record structure. For example, the following statement transfers a file named C205B to a NOS/BE system:

```
MFLINK,C205B,ST=XXX,DD=US,JCS="CATALOG,C205B,ID=XX,RP=90."
```

3. Execute the CYBER 170 program that uses the data. To read records from the data file, it should use a BUFFER IN statement that reads a record of length LEN1. (An unformatted READ statement could be used instead if the job includes a FILE control statement that specifies BT=C,RT=S.)

After executing the BUFFER IN statement, the data is available for processing by the program.

TAPE LABELS AND FORMATS

F

VSOS supports ANSI standard tape labeling (level 2, ANSI standard X3.27-1978).

Tape labels delimit and identify the data recorded on a tape volume. When the label type specified for a tape file is ANSI standard labeled, VSOS writes and expects to read the following required labels:

<u>Label</u>	<u>Description</u>
VOL1	Marks the beginning of a tape volume.
HDR1	Marks the beginning of a tape file.
EOF1	Marks the end of a tape file.
EOV1	Marks the end of a tape volume (used only if the end of the volume precedes the end of the file).

A user can also specify the following additional optional labels:

<u>Label</u>	<u>Description</u>
UVLn	Sequence of one to nine additional volume labels (n is a digit, 1 through 9).
HDRn	Sequence of one to eight additional header labels (n is a digit, 2 through 9).
UHLa	Sequence of additional header labels (a can be any character).
EOFn	Sequence of one to eight additional end of file labels (n is a digit, 2 through 9).
UTLa	Sequence of additional trailer labels (a can be any character).

For more information on specifying additional optional labels, refer to the Q5OPEN and Q5REELSW call descriptions.

Figure F-1 shows ANSI standard tape label groupings. As shown, multifile sets require ANSI standard labeling to delimit the files in the set.

Unlabeled tapes using I, SI, or LB tape format also use EOF1 and EOV1 labels to mark the end of the file data as shown in figure F-2. V format unlabeled tapes use no labels.

Each tape label is written as an 80-character PRU. Table F-1 lists the formats of the required ANSI labels; table F-2 lists the formats of the optional labels.

Tables F-1 and F-2 list each field, its content, and the default value VSOS writes in the field if no other value is specified for the field. The figures also indicate whether VSOS checks the existing field value when it reads or writes the label. The phrase, checked if specified, indicates that VSOS checks that the existing field value matches the value specified for the field only when a value is specified.

For more information on specifying label values, refer to the LABEL control statement and Q5LABEL call descriptions in chapters 4 and 9 of this manual.

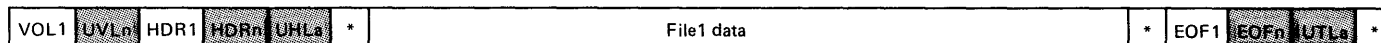
Table F-3 describes tape group separators for each label type and tape format. In addition, the allowance of multivolume and multifile processing is shown.

Figure F-3 provides a summary of the physical data layout on the tape media for supported record type/tape format/tape blocking combinations.

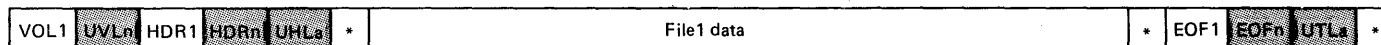
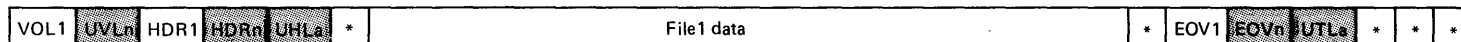
Figure F-4 describes the layout of labels on the tape media for tape labeling/tape format combinations that are supported on single-volume tape. Figure F-5 is similar for multivolume tape.

(Note: Shading indicates that the label represents a sequence of optional user-specified labels; an asterisk represents tape mark):
 An ANSI labeled NV format tape can also have tape marks embedded within the file data. The tape marks act as group delimiters.

Single File on Single Tape Volume



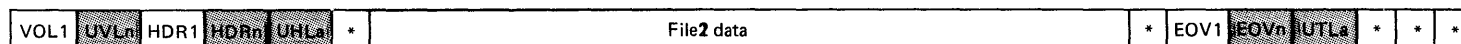
Multivolume File



Multifile Set on a Single Volume



Multifile Set on Three Volumes



NV Format

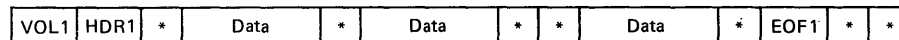


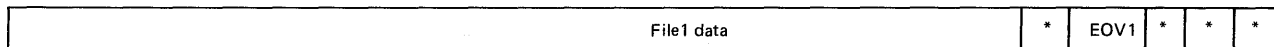
Figure F-1. ANSI Standard Tape Label Groupings

(Note: An asterisk represents a tape mark):

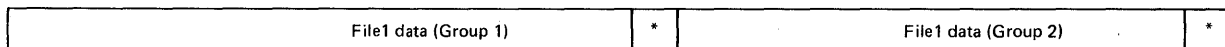
Single Volume File Using I, SI, or LB Format



Multivolume File Using I, SI, or LB Format



Single Volume File Using V or NV Format



Multivolume File Using V or NV Format

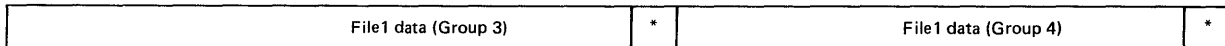
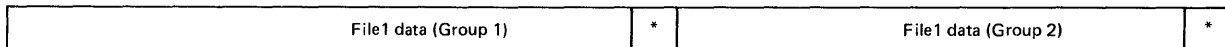


Figure F-2. Unlabeled Tape Files

Table F-1. Required ANSI Label Formats (Sheet 1 of 4)

Byte	Field	Content	Default Value Written	Compared on Read (LPROC=R)	Compared on Write (LPROC=W)
VOL1 Label					
1-3	Label identifier	VOL	VOL	Yes	Yes
4	Label number	1	1	Yes	Yes
5-10	Volume identifier	Volume serial number (VSN)	As read from existing label	Yes, if file assigned by VSN	Yes, if file assigned by VSN
11	Accessibility	Character restricting access to the volume	Blank (unlimited access)	Yes	Yes (must be specified on the OVA parameter)
12-37	Reserved for future standardization	Blanks	Blanks	No	No
38-51	Owner identifier	Characters identifying the owner	Blanks	No	No
52-79	Reserved for future standardization	Blanks	Blanks	No	No
80	Label standard version	3 indicates conformity with ANSI X3.27-1978 standard. Blank indicates agreement between interchange parties required.	3	Yes	Yes
HDR1 Label					
1-3	Label identifier	HDR	HDR	Yes	Yes
4	Label number	1	1	Yes	Yes
5-21	File identifier	1 to 17 characters identifying the file	File name as specified on label specification	Checked if specified	Checked if specified

Table F-1. Required ANSI Label Formats (Sheet 2 of 4)

Byte	Field	Content	Default Value Written	Compared on Read (LPROC=R)	Compared on Write (LPROC=W)
22-27	File set identifier	One to six characters identifying the multifile set (not currently supported by VSOS)	Blanks	No	No
28-31	File section number	Four digits identifying the file section: 0001 for first HDR1 label of file; incremented for each subsequent HDR1 label of file. Used when positioning a file using a saved tapes table entry.	0001	No	No
32-35	File sequence number	Four digits identifying the position of the file within the multifile set. If 9999 specified, next number in sequence assigned to field.	0001	Checked if specified	Checked if specified
36-39	Generation number	Four digits identifying the generation of the file.	0001	No	No
40-41	Generation version number	Two digits identifying the version of the file generation.	00	No	No
42-47	Creation date	Six characters: a blank, two digits indicating the year, and three digits indicating the day of the year.	Space followed by five zeros.	No	No
48-53	Expiration date	Six characters: a blank, two digits indicating the year, and three digits indicating the day of the year.	Space followed by five zeros.	Checked if file opened for write access	Yes

Table F-1. Required ANSI Label Formats (Sheet 3 of 4)

Byte	Field	Content	Default Value Written	Compared on Read (LPROC=R)	Compared on Write (LPROC=W)
54	Accessibility	Character restricting access to the file. If character is A, the user number of the task must match the owner identifier of the volume	Blank (unlimited access)	Yes	Yes (must be specified on the OFA parameter)
55-60	Block count	Must be zeros.	Zeros	No	No
61-73	System code	1 to 13 characters identifying the system that wrote the label.	CYBER 200 2.3	No	No
74-80	Reserved for standardization	Blanks	Blanks	No	No
EOF1 Label					
1-3	Label identifier	EOF	EOF	Yes	No, new label always written
4	Label number	1	1	Yes	No, new label always written
5-54	Same as corresponding HDR1 label	Same as corresponding HDR1 label	Same as corresponding HDR1 label	No	No, new label always written
55-60	Block count	Six digits that indicate the number of data blocks in the file	Blanks	Yes	No, new label always written
61-80	Same as corresponding HDR1 label	Same as corresponding HDR1 label	Same as corresponding HDR1 label	No	No, new label always written

Table F-1. Required ANSI Label Formats (Sheet 4 of 4)

Byte	Field	Content	Default Value Written	Compared on Read (LPROC=R)	Compared on Write (LPROC=W)
EOV1 Label					
1-3	Label identifier	EOV	EOV	Yes	No, new label always written
4	Label number	1	1	Yes	No, new label always written
5-54	Same as corresponding HDR1 label	Same as corresponding HDR1 label	Same as corresponding HDR1 label	No	No, new label always written
55-60	Block count	Six digits that indicate the number of data blocks written since the last HDR1 label on the volume	Blanks	Yes	No, new label always written
61-80	Same as corresponding HDR1 label	Same as corresponding HDR1 label	Same as corresponding HDR1 label	No	No, new label always written

Table F-2. Optional Label Formats (Sheet 1 of 2)

Byte	Field	Content	Default Value Written	Compared on Read (LPROC=R)	Compared on Write (LPROC=W)
HDR2 Through HDR9 Labels					
1-3	Label identifier	HDR	None	Yes	No
4	Label number	Digit in the range 2 through 9	None	Yes	No
5-80	User option	Any characters	None	No	No

Table F-2. Optional Label Formats (Sheet 2 of 2)

Byte	Field	Content	Default Value Written	Compared on Read (LPROC=R)	Compared on Write (LPROC=W)
EOF2 Through EOF9 Labels					
1-3	Label identifier	EOF	None	Yes	No
4	Label number	Digit in the range 2 through 9	None	Yes	No
5-80	User option	Any characters	None	No	No
EOV2 Through EOV9 Labels					
1-3	Label identifier	EOV	None	Yes	No
4	Label number	Digit in the range 2 through 9	None	Yes	No
5-80	User option	Any characters	None	No	No
UVLn Labels					
1-3	Label identifier	UVL	None	Yes	No
4	Label number	Digit identifying the position of the label in the sequence of UVL labels	None	Yes	No
5-80	User option	Any characters	None	No	No
UHLA and UTLA Labels					
1-3	Label identifier	UHL or UVL	None	Yes	No
4	Label number	Any character	None	Yes	No
5-80	User option	Any character	None	No	No

Table F-3. Tape Group Separators

Label Type	Tape Format	Multi-volume	Multi-file	Multigroups Per File			
				Rec Type B	Rec Type W, L	Rec Type R	Rec Type F, V
AN	I,SI,LB	YES	Y (1)	Y (17B)	Y (CW)	Y (GS)	NO (2)
UL	I,SI,LB	YES	NO	Y (17B)	Y (CW)	Y (GS)	NO (2)
AN	V	YES	Y (1)	NO	Y (CW)	Y (GS)	NO
AN	NV	YES	Y (1)	Y (*)	Y (CW)	Y (GS)	NO (2)
UL	V,NV	YES	NO	Y (*)	Y (CW)	Y (GS)	NO (2)

- (1) Tape marks with adjacent labels delimit files on multifile tapes. Positioning to a file is via LABEL statement only.
- (2) Multigroup F records are not defined for tape or disk. A level 17 or tape mark will be ignored if read from tape.
- (17B) Level 17 zero-length PRU is group delimiter for record type B on I, SI, or LB-format tape.
- (*) Hardware tape mark is group delimiter for record type B on V or NV-format tape.
- (CW) The end-of-group W-control word at end of LRU delimits groups for W-type records; the end-of-partition W-control word at end of LRU delimits groups of L-type records.
- (GS) The ASCII group separator (hex 1D) at end of LRU delimits groups for R-type records (if RMK is VS or RS [hex 1F or 1E]).

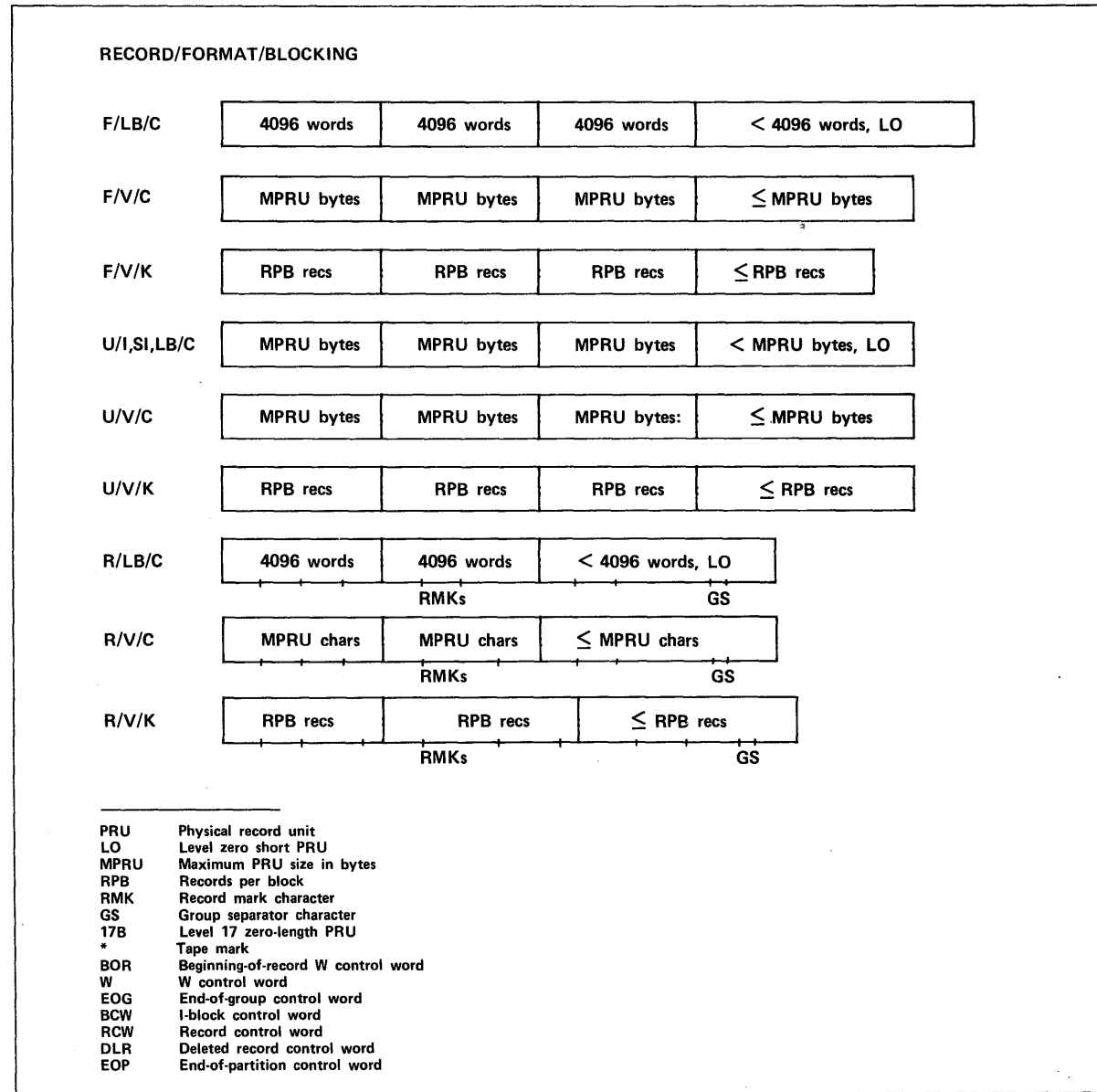


Figure F-3. Summary of Tape Blocks per Group (Sheet 1 of 2)

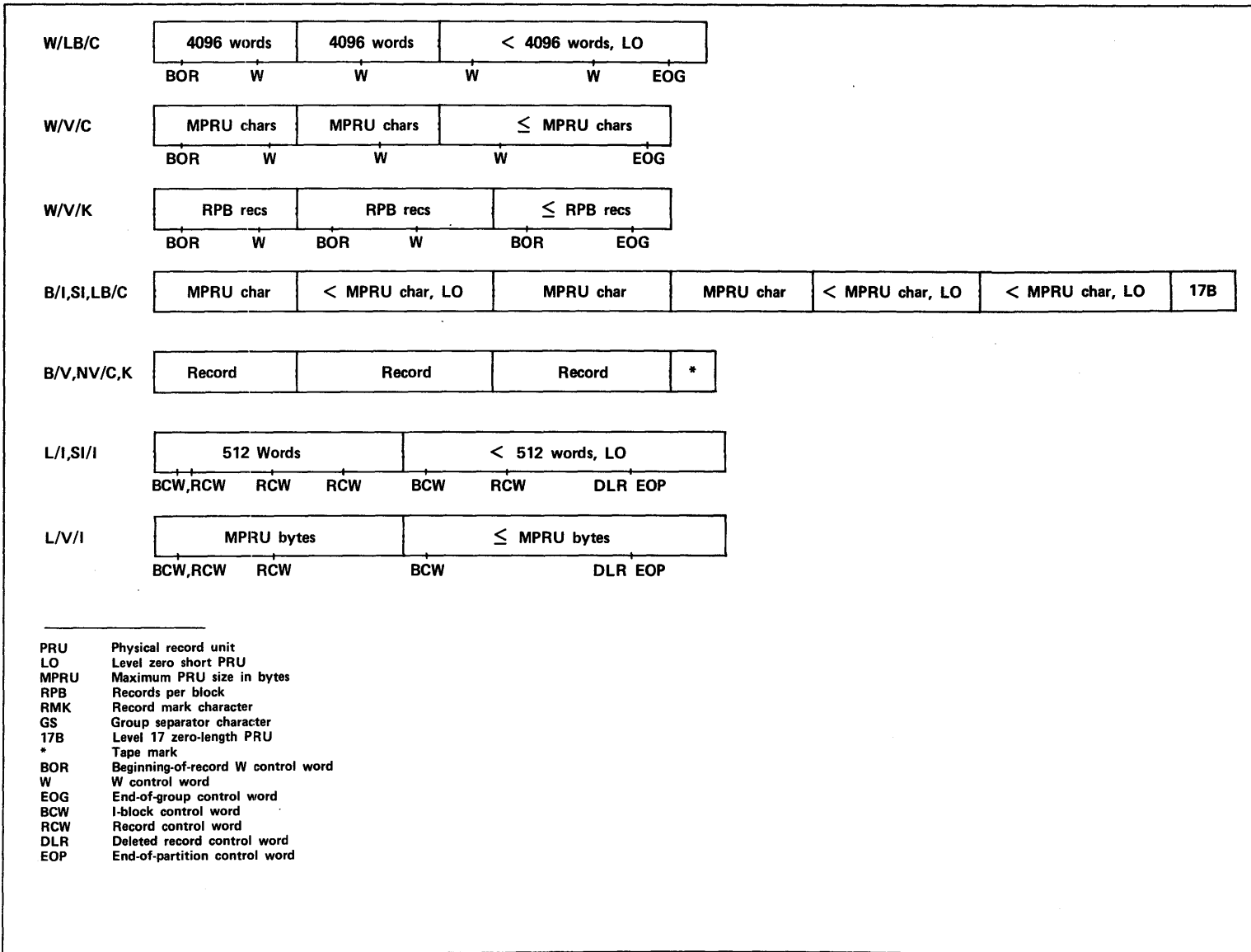
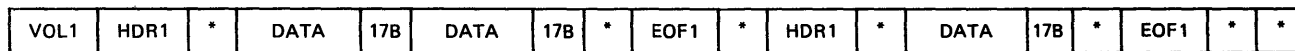


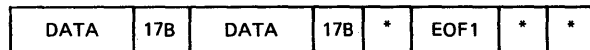
Figure F-3. Summary of Tape Blocks per Group (Sheet 2 of 2)

Examples of the currently defined tape label types AN and UL for all tape formats are shown in Figure G-4.
 (* represents a tape mark.)

1. AN LABELED I,SI,LB



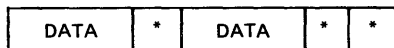
2. UL UNLABELED I,SI,LB



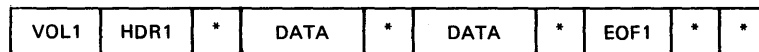
3. AN LABELED V-FORMAT



4. UL UNLABELED V-FORMAT



5. AN LABELED NV-FORMAT



6. UL UNLABELED NV FORMAT

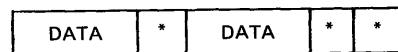


Figure F-4. Single Volume Tapes

Examples of the currently defined tape label types AN and UL for all tape formats are shown in figure F-5 (* represents a tape mark).

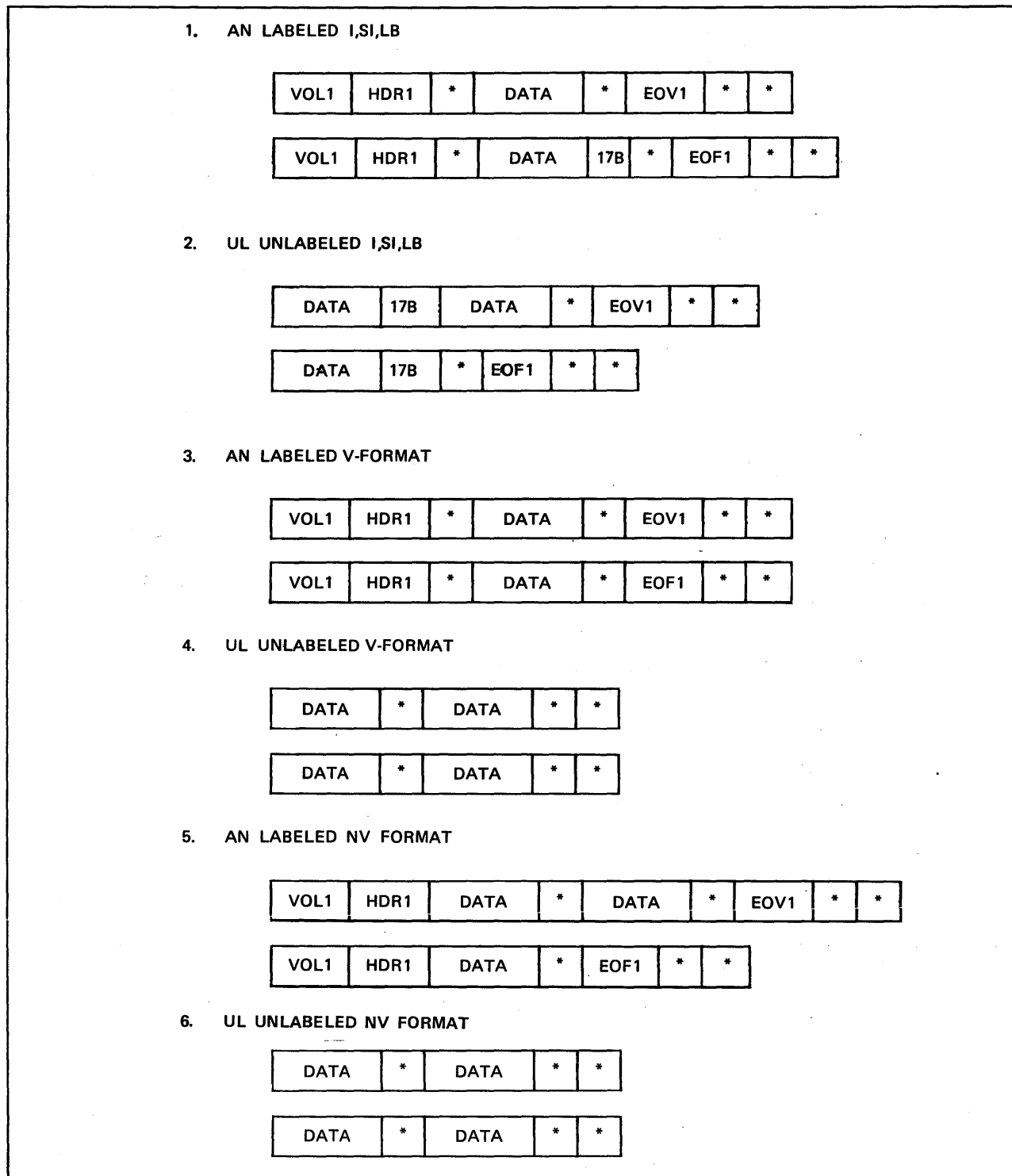


Figure F-5. Multivolume Tapes (Sheet 1 of 2)

DISPLAY PROGRAM EXAMPLE

G

The following FTN200 program illustrates many of the functions of Q9SCR. When run, the program should draw a box at the outer edges of the terminal screen, then draw another box inset within that box. Within this inner box, first the word "OKAY" should appear in two steps, and then the numbers 0 to 20 should appear, one per line. Since the window is less than 21 lines tall, the window will scroll as the later numbers are displayed. This sequence should repeat. Finally, the cursor will be positioned near the bottom, left corner, where the "STOP" and "ALL DONE" messages will appear.

```
program test(output, tape6=output)
implicit integer (a-z)

*   define bounds of windows.  first, the outer window

parameter (yposbox = 0)
parameter (xposbox = 0)
parameter (ybox = 24)
parameter (xbox = 80)

*   next, a subwindow of that window

parameter (ypossub = 2)
parameter (xpossub = 10)
parameter (ysbox = ybox - ypossub - 5)
parameter (xsbox = xbox - 2*xpossub)

*   and a scrolling window within the limits of the subwindow

parameter (yposscr = ypossub + 1)
parameter (xposscr = xpossub + 1)
parameter (yscr = ysbox - 2)
parameter (xscr = xsbox - 2)

real boxing, subbox, owlwin, result, scrwin
real q9scr

parameter (bufsize = 1024)
common buff(bufsize, msgbuf(300))

*   get the function selector and option numbers

addstr = q9scr(1, 'addstr')
if (addstr .eq. 0) then
  stop 'cannot look up function names'
endif
box = q9scr(1, 'box')
clear = q9scr(1, 'clear')
delwin = q9scr(1, 'delwin')
endscr = q9scr(1, 'endscr')
initscr = q9scr(1, 'initscr')
overlay = q9scr(1, 'overlay')
newwin = q9scr(1, 'newwin')
```

```

printw = q9scr(1,`printw`)
refresh = q9scr(1,`refresh`)
reset = q9scr(1,`reset`)
scrlok = q9scr(1,`scrollok`)
subwin = q9scr(1,`subwin`)
mv = q9scr(1,`move`)
w = q9scr(1,`wind`)

wbox = w + box
mvadds = mv + addstr
wdelwin = w + delwin
wmvadds = w + mv + addstr
wmove = w + mv
wovrlay = w + overlay
wrfresh = w + refresh

* change the next line to match the terminal type used for testing

termtyp = q9scr(1,`il100`)

* initialize screens and get back a pointer to the basic window

boxing = q9scr(initscr, buff, bufsize, 1, 0, ybox, xbox)
call check(boxing, `initscr`)
if (boxing .eq. 0) then
    stop `init failed, giving up`
endif

do 9000 pass = 1, 2

* define the subwindow

subbox = q9scr(subwin, ysbox, xsbox, ypossub, xpossub)
call check(subbox, `subwin subbox`)

* define the window for testing the "overlay" function

ovlwin = q9scr(newwin, ybox, xbox, yposbox, xposbox)
call check(ovlwin, `newwin ovlwin`)

* draw a border for the basic window and put in some text containing
* spaces

result = q9scr(wbox, boxing, `$`, ` `)
call cehck(result, `wbox`)

result = q9scr(mvadds, ypossub+1, xpossub+1, `o a`)
call check(result, `mvadds`)

result = q9scr(refresh, msgbuf, termtyp)
call check(result, `refresh`)
if (result .gt. 0) then
    call q5ndmjc(`msg=`,msgbuf,`len=`,result,`reject`)
endif

* draw a border for the subwindow

result = q9scr(wbox, subbox, `.` , `.`)

```

```

call check(result, `wbox`)

* put some text in the overlay window synchronized to the text in the
* basic window, then overlay that text on the basic window

result = q9scr(wmvadds, ovlwin, yposs+1, xposs+1, ` k y`)
call check(result, `wmvadds`)

result = q9scr(wovrlay, boxing, ovlwin)
call check(result, `wovrlay`)

result = q9scr(refresh, msgbuf, termtyp)
call check(result, `refresh`)
if (result .gt. 0) then
    call q5sndmjc(`msg=`,msgbuf,`len=`,result,`reject`)
endif

* delete the overlay window and create the scrolling window

result = q9scr(wdelwin, ovlwin)
call check(result, `delwin`)

sclwin = q9scr(newwin, yscr, xscr, yposs, xposs)
call check(sclwin, `newwin sclwin`)

result = q9scr(scrlok, 1)
call check(result, `scroll ok`)

* print enough strings to the window to force it to scroll

do 1000 y = 0, yscr + 5

    result = q9scr(printw, `%n->%d<-`, y)
    call check(result, `printw`)

    result = q9scr(refresh, msgbuf, termtyp)
    call check(result, `scroll refresh`)
    if (result .gt. 0) then
        call q5sndmjc(`msg=`,msgbuf,`len=`,result,`reject`)
    endif
1000 continue

* the first time through, discard all the window structures

if (pass .eq. 1) then

    result = q9scr(endscr)
    call check(result, `endscr`)

* then try to reinitialize them via reset

    boxing = q9scr(reset, 1)
    call check(boxing, "reset")
endif
9000 continue

* move the cursor to a nice place before exiting

```

```

result = q9scr(wmove, boxing, ybox-3, 0)
call check(result, 'wmove')

result = q9scr(refresh, magbuf, termtyp)
call check(result, 'refresh')
if (result .gt. 0) then
  call q5sndmjc('msg=',msgbuf,'len=',result,'reject')
endif

stop
end
subroutine check(rslt, w)

```

* subroutine to report any errors returned by Q9SCR

```

real rslt
character*(*) w
real astext
integer errc
data errc/5/

if (rslt .eq. 0.0) then
  astext = q9scr(1, rslt)
  if (astext .eq. 0.0) astext = 8hunknown
  print 700, rslt, astext, w

700  format(' got back ',z16,' (',a8,') for ',a)
  errc = errc - 1
  if (errc .le. 0) then
    stop 'excessive errors'
  endif
endif
return
end

```

INDEX

- Abnormal job termination 3-18
 - EXIT statement 4-48
- Abnormal termination C-1
- Abnormal termination control (ATC) 3-29
 - Enabling/disabling 3-29
 - Subroutine 3-29
- Abort 3-18
 - Dump 3-18
 - Flag 3-18
- Access C-1
 - Modes 2-14
 - Permission 2-14
 - Listing 4-61
 - Sets 2-14
- Access permission
 - Changing 4-103
- Accessing tapes table entry 9-37
- Accessing the system 3-1
- Account block C-1
- Account identifier 3-4; C-1
 - Validation 3-4
- Account number 4-21; C-1
- Accounting 3-33
- ADDFILE Update directive 5-15
- Addressing 1-6
- Advanced Tape System (ATS) 1-3
- Advising the system of task virtual space 8-8
- Allocating static stack memory space 8-73
- Allocation unit C-1
- Alter a task's attributes 4-139
- @ and ^ character removal 4-11
 - Suppressing 4-12
- ANSI
 - Carriage control 2-5
 - Fixed length record format 2-18
 - Standard labeled tape files 2-30
 - Standard labels 2-30
- ANSI standard tape labeling F-1
- Append access permission 2-14
- Appending data 9-7
 - Blocks 9-130
 - Records 9-98
- Application programs 1-5
- Archive file format 4-41
- Archiving files 4-36.2
- Archiving to a front-end system 4-42
- Archiving to CYBER 200 mass storage 4-42
- Archiving to CYBER 200 on-line tapes 4-43
- Arithmetic conversion routines E-5
- ASCII
 - Carriage control 2-5
 - Character set A-1
 - Debug directive 6-8
- Assembly/disassembly option 3-28
- Assembly/disassembly option tape 2-32
- Assign account and project number 4-21
- ATC (abnormal termination control) 3-29
 - Enabling/disabling 3-29
 - Subroutine 3-29
- ATS (Advanced Tape System) 1-3
- ATTACH control statement 4-13
 - WAIT parameter 4-13
- Attached pools 3-8
- Attaching
 - Permanent files
 - Control statement 4-13
 - Program call 9-10
 - Pools
 - Control statement 4-99
 - Program call 9-85
- Attributes
 - Change 4-139
 - Files 2-1
- AUDIT control statement 4-15
- AUDIT file specification 4-15
- AUDIT output listings 4-20
- AUDIT request 3-25
- Autoload C-1
- Automatic volume recognition 2-29
- Available permissions 3-4
- B record format 2-22
- BACK
 - Debug directive 6-6
 - Look directive 6-15
- Bank update table 8-68
- BASE Look directive 6-15
- Batch
 - Deck C-1
 - Execute line 3-14
 - Input file 3-6; C-1
 - Structure 3-14
 - Job 3-14; C-1
 - Processing 3-16

- Scheduling 3-15
- Processor 3-14; C-1
- Control statements 4-1
- System access 3-6
- Batch input file
 - Rerun 8-78
- Batch resource limits 3-31
- BATCHPRO (batch processor) C-1
- BB request line 3-8
- BEFORE Update directive 5-16
- BEGIN statement 4-6.3
- Beginning of information (BOI) C-1
- BINARY file 3-19
- BIT Look directive 6-15
- BKPT Debug directive 6-10
- BKPTR Debug directive 6-10.1
- Blank common 4-72
- BLANK control statement 2-31
- Block C-1
 - Grouping unmapped 4-72
 - I/O 9-7
 - Reading 9-102
- Block count discrepancy 2-31
- Blocking types 2-23,33
- BOI (beginning of information) C-1
- Bound explicit and implicit maps 2-4
- BREAK character 3-7
- ! break character 3-7
- Buffer record 9-39
- Buffered I/O 2-17
- Buffers
 - I/O 9-82.1
- BYE request line 3-10
- Byte C-2

- C blocking 2-23
- Call format 8-6
- Call parameters 5-4
- CALL Update directive 5-16
- Calling parameters 8-7
- Card identification 5-10
- Card image file 5-1
- Carriage control 2-5
- Case conversion request 3-9
- Category 3-15,33
- Change attributes 4-139
- Change job characteristics 4-127
- Change request 3-7
- Changing
 - Access permission set
 - Control statement 4-103
 - Program call 9-89
 - Accounting rate 8-96
 - File attributes
 - Control statement 4-139
 - Program call 9-12

- File ownership
 - Control statement 4-53
 - Program call 9-60
- FIT field values 9-122
- Job memory limits 4-127
- Large page limit
 - Control statement 4-127
 - Program call 8-79
- Object module linkages 4-69
- Password 4-98
- Working set size limit for job 4-127
- Channel 1-3
- Character code conversion 4-90
- Character count blocking 2-23
- Character set 1-5
- CHARGE control statement 4-22
- Charge number C-2
- Check block I/O request status 9-19
- Checking
 - I/O request status 9-16
 - Termination values 4-140
- Checkpoint C-2
- Checkpointing a task 7-1
- CHKPNT call 7-1
- Clear tape I/O error 9-21
- Closing files 9-22
- Coded tape file 2-32
- COMDECK Update directive 5-17
- COMMENT control statement 4-22
- Comments
 - In job dayfile 4-22
 - In Update deck 5-25
 - On control statement 3-15
- \$ common block 4-69
- Common blocks 4-69
- COMPARE control statement 4-23
- Comparing controllee file 4-24
- Comparing file contents 4-23
- Comparison 4-140
- Compile file 5-8
- COMPILE Update directive 5-18
- Concatenating substitution values 4-11
- Concurrent file access 2-16
- Conditional breakpoints 6-10.1,11,12
- Configuration 1-1
- Connected interactive terminal files 2-37
- Construct system shared library 4-129
- & continuation character 4-5
- CONTINUE Debug directive 6-10.1
- Control statement 4-1; C-2
 - Batch execution 3-14
 - Condition testing 4-6.1
 - Conditional 4-6.1
 - Processing 4-6.2
 - Execution sequence 4-7
 - Format 4-74.1 (also refer to
 - Execute line)
 - Group 3-14

- Interactive execution 4-5
- List of 4-1
- Management 4-6
- Parameter format 4-4
- Procedures 4-6.3
- Summary 4-1
- Variables 4-6
- Control word delimited (W) record format 2-20
- Controllee 8-55,80,92; C-2
 - Chain 3-3; 8-56,69; C-2
 - Execution 3-1,12
 - File 2-2; 3-1; C-2
 - Formats 4-66
 - Software C-2
 - Termination status
 - Returning 8-40
 - Setting 8-93
- Controllees
 - Dynamic linking 4-70
- Controller 8-82; C-2
 - Hardware 1-2
 - Software 3-1
- Conversion
 - Example E-8
 - Logical structures 4-90
- Conversion mode
 - CM parameter 2-32
- Conversion routines E-1
- Convert parameter 2-32
- Copy calls 8-41
- COPY control statement 4-25
- Copying 8-41
 - Bank update table 8-68
 - File 4-25
 - File indices
 - By hierarchical search 8-58
 - Pool 8-60
 - Private 8-63
 - Public 8-66
 - Invisible package 8-41
 - Logical records 4-28
 - Minus page 8-48
 - Pack label and file indices 8-49
 - Register file 8-42
 - Statistics buffer 8-71
 - Timecard buffer 8-72
- Copying controllee file 4-27
- Copying dayfile 4-30
- Copying to or from a mass storage file 4-26
- Copying to or from a tape file 4-26
- COPYL control statement 4-28
- Correction run 5-3,11
- CPU 1-6; C-2
 - Scheduler 3-15
- Create local file 9-115
- Creating
 - Controllee files 4-65
- Library files 4-94
- Local files 4-112
- Modmerge files 4-94
- Permanent files
 - Control statement 4-30
 - Program call 9-25
- Pool files
 - Control statement 4-54
 - Program call 9-60
- Pools
 - Control statement 4-100
 - Program call 9-86
- Public files
 - Control statement 4-54
 - Program call 9-60
- Creation of library or modmerge file 4-94
- Creation run 5-2,9
- CYBER Record Manager control word (L) record format 2-21
- CYBER 170 arithmetic conversion routines E-5
- CYBER 170 compatibility 2-32
- CYBER 170 tape interchange 2-21
- CYBER 18 MCU 1-2
- CYBER 200 archiving 3-27
- CYBER 200/CYBER 170 comparison 1-5
- CYBER 200 job submission 3-22
- CYBER 200 model descriptions 1-2
- CYBER 200 RHF (see also RHF) 3-6
- Data
 - Conversion options 2-32
 - File 2-2; C-2
 - Flag Branch Manager routines 3-28
 - Input 3-12
 - Data base C-2
 - Data file C-2
 - Data format specifications 4-119
 - DATAOUT file 3-20
 - DAU (device allocation unit) 2-27; C-2
 - Dayfile 3-16; C-2
 - Copying 4-30
 - DAYFILE control statement 4-30
 - DDECIMAL Debug directive 6-5
 - DEBUG
 - Control statement 6-2
 - Directives 6-3
 - Use at security-sensitive sites 6-2
 - Debugging utilities 6-1
 - Debugging versions of routines 4-70
 - DEC Look directive 6-15
 - DECIMAL Debug directive 6-8
 - Decimal to hexadecimal conversion A-6
 - Deck 5-1
 - DECK Update directive 5-19
 - Decoding
 - Disk status table 8-11

Miscellaneous table 8-12
 Pack label 8-29
 Permanent file indices 8-18
 Default LOAD library 4-65
 Default project number C-2
 DEFINE control statement 4-30.1
 DEFINE Update directive 5-20
 DELETE Update directive 5-20
 Density for recording tapes 2-32
 Descriptor parameter 8-6
 Destroying
 Batch input file
 At job termination 3-18
 Program call 8-31
 Permanent files
 Control statement 4-106.1
 Program call 9-94
 Pool files
 Control statement 4-102
 Program call 9-94
 Pools
 Control statement 4-102
 Program call 9-87
 Public files
 Control statement 4-46
 Program call 9-94
 Detaching
 Permanent files
 Control statement 4-124
 Program call 9-106
 Pools
 Control statement 4-102
 Program call 9-88
 Device allocation unit (DAU) 2-27; C-2
 Device characteristics 2-26
 Device number C-2
 Device overflow C-3
 Device set 2-26; C-3
 Device types 2-23
 DFLOAT Debug directive 6-5
 Diagnostic messages B-1
 Direct access file organization 2-25
 Direct access file transfers 3-25
 Directive(s) 5-14; C-3
 Directory file 4-37
 Disabling
 Abnormal termination control 3-29
 Program call 8-33
 Message interrupts 8-32
 User reprieve 8-75
 Disabling standard error recovery 2-37
 Discarding FITs 9-109
 Disk
 Blocks 2-26
 Files 2-1
 Status table 8-11
 Units 1-2
 DISPLAY
 Debug directive 6-5

Look directive 6-15
 Disposition code 2-4
 DIVERT control statement 4-34
 DLF (Dump/Load Facility) 3-21,27
 Archiving 4-36.2
 DMAP control statement 4-34.2
 DREG Debug directive 6-7
 DROP control statement 4-36
 Drop file 2-2; C-3
 Naming 2-3
 DUMP control statement 6-23
 Dump/Load Facility (DLF) 3-21,27
 Archiving 4-36.2
 DUMPF control statement 4-36.2
 Output 4-43
 Example 4-45
 Dumping
 Cumulative accounting file 8-34
 Drop file information 6-23
 Permanent files 4-36.2
 DVSTnn C-3
 Dynamic
 Allocation C-3
 Execution 3-12; 4-70
 Library C-3
 Linking 4-70; C-3
 Loading C-3
 Module 4-70; C-3
 Stack address 4-69

 EASCII Look directive 6-15
 EDEC Look directive 6-15
 EDITPUB control statement 4-46
 EFLOAT Look directive 6-15
 EHDEC Look directive 6-15
 EHEX Look directive 6-15
 EHFLOAT Look directive 6-15
 ELSE control statement 4-6.2
 Enabling
 Abnormal termination control 3-29
 Program call 8-37
 Message interrupts 8-35
 User reprieve 8-75
 END
 Debug directive 6-ii
 Look directive 6-15
 End of file (EOF) C-3
 End of information (EOI) C-3
 End of partition codes 9-57
 End of tape processing 2-29
 ENDIF control statement 4-6.2
 ENDIF Update directive 5-21
 EOF (end of file) C-3
 EOF1 label 2-30,37
 EOI (end of information) C-3
 EOV1 label 2-30,37
 EREG Debug directive 6-7
 Error codes

CHKPNT 7-3
 Error conditions 2-7
 Error message(s) B-1
 Format B-1
 Routing 8-5
 System utility B-26
 Error processing 8-5
 SIL 8-5
 System 3-20
 Error recovery
 Codes 4-88
 Limit 3-30
 Tape 2-37
 Error severity
 SIL 8-5
 System 3-18
 Error threshold value 3-16
 Errors B-1
 ETP parameter 2-29
 Evicting local files 4-124
 Examining a mass storage file 6-13
 Executable file 2-2
 Generation 4-65
 Execute access permission 2-15
 EXECUTE Debug directive 6-10.1
 Execute line 3-10.1
 Batch 3-14
 Interactive 3-10.1
 Executing task status 4-109
 Execution
 Dynamic 4-71
 Execution line format 3-11
 Execution time file reassignment 8-80
 EXIT control statement 3-18; 4-48
 Exit path 4-48
 Exit processing 3-18
 Explicit file routing 3-23; 4-91
 Explicit I/O 2-17,25; 9-4; C-3
 Using SIL calls 9-2
 External references 4-69

 F format records 2-25
 F record format 2-18
 Family C-3
 Family of print files 2-5
 Fatal SIL errors 8-5
 File 2-1,28,37; C-4
 Access permissions 2-14
 Archiving 3-27; 4-36.2,40.1
 Blocking 2-23
 Connected to terminal 2-37
 Creation 2-27
 Duration 2-8
 Extensions 2-28
 Families 2-5; 3-18
 Formats 2-18
 I/O 2-4,17
 Identifier (HDR1) 2-30
 Index table (FILEI) C-4
 Information 4-15
 Information table (FIT) 2-37; 9-3; C-4
 Format D-1
 Limit 9-16
 Logical unit number 9-4
 Mapping 2-27
 Naming conventions 2-1
 Organization 2-25
 Ownership 2-9
 Patterning 2-9
 R formatted 2-19
 Reloading 3-24; 4-80,84,85
 Routing 4-91
 Program call 9-112
 Search hierarchy 2-12
 Security levels 2-9
 Segments 2-26
 Size 2-26
 Space allocation 2-26
 Structure 2-18
 Types 2-26; C-4
 Usage controls 2-9
 At security-sensitive sites 2-9
 FILEI (file index table) C-4
 FILES control statement 4-49
 FILES output 4-51
 FIT (file information table) 2-37; 9-3; C-4
 Format D-1
 Limit 9-16
 Fixed-length
 Blocking 2-24
 Record format 2-18
 Fixed length records 2-25
 FLOAT
 Debug directive 6-8
 Look directive 6-15
 flun (file logical unit number) 9-4
 Formal parameter substitution 4-8,9
 Formats 2-18
 Tapes 6-1
 FORTRAN compiler execution 3-19
 FORTRAN data conversion routines E-1
 Front-end processor 1-3
 Full length PRU C-4
 Full update mode 5-7

 General access permission set 2-15
 Generating a FIT (file information table) 9-31
 Getting
 Accounting information
 At terminal 3-8
 Within program 8-38
 Current date and time
 At terminal 3-8
 Within program 8-95
 FIT field values 9-50

Large page limits 8-43
 Message from
 Controllee 8-44
 Controller 8-45
 Operator 8-47
 Program state 3-8
 Task
 Information 8-51
 Time limit 8-50
 User number 8-53
 Time task has used
 At terminal 3-8
 Within program 8-10
 GIVE control statement 4-53
 GO file 3-20
 Granting access to a pool
 Control statement 4-97
 Program call 9-91
 Group 2-18; C-4
 Group controllee file blocks 4-72
 Grouping data with code 4-69
 Grouping parameter mapping 4-73
 Grouping parameters on LOAD 4-69
 Grouping unmapped blocks 4-72

 Hardware description 1-1
 HDEC Look directive 6-15
 HDR1 label 2-30; 4-59
 HELLO request line 3-10
 HEX Debug directive 6-8
 HEX Look directive 6-15
 # hexadecimal character 4-4
 Hexadecimal number system 1-5
 Hexadecimal to decimal conversion A-6
 Hexadecimal to octal conversion A-5
 HFLOAT Look directive 6-15

 I blocking 2-23
 I/O 9-3,5
 Buffer overlap 9-104
 Buffers 2-17; 9-82.1
 Calls 9-9
 Overview 9-3
 I/O calls 9-1
 I request line 3-9
 I tape format 2-34
 IBG (interblock gap) 2-23; C-4
 IBM arithmetic conversion routines E-1
 IDEC Look directive 6-15
 IDENT Update directive 5-20
 IDISPLAY Debug directive 6-5
 IDREG Debug directive 6-7
 IF control statement 4-6.1,6.2
 IF Update directive 5-22
 IFLOAT Look directive 6-15
 IHDEC Look directive 6-15

 HSEARCH Look directive 6-17
 IHEX Look directive 6-15
 IHFLOAT Look directive 6-15
 Implicit I/O 1-2; 2-17; 9-8; C-4
 Individual access permission sets 2-14
 Information requests 3-8
 Initialization 4-66,73
 Initiation of controllee execution 3-1
 INPUT file 3-14,16,19
 Input line format 3-6
 Input/output channels 1-3
 Input/output connector (IOC) C-4
 Input queue 3-14
 Input queue manager (IQM) 1-4; 3-6; C-4
 Input queue status 4-108.1
 INSERT Update directive 5-21
 Integer conversion E-5
 Interactive access 3-5
 Via RHF 3-21
 Interactive execute line 3-10.1
 Interactive execution of FILE utility 4-52
 Interactive resource limits 3-32
 Interactive session C-4
 Interactive tape usage 2-28
 Interactive Transfer Facility Servicer
 (ITFS) 1-4; 3-21
 Interblock gap (IBG) 2-23; C-4
 Interface language 1-5
 Internal blocking 2-23
 Internal characteristic 2-8
 Interrupt mode disable 3-30
 Interrupt request 3-7
 Interrupt subroutine 3-29; 8-77
 INTRACTV job category 3-32
 Invisible package C-4
 IOC (input/output connector) C-4
 IQM (input queue manager) 1-4; 3-6; C-4
 ITFS (Interactive Transfer Facility
 Servicer) 1-4; 3-21

 JDEFAULT job category 3-31
 JDN (job descriptor number) 3-5,7,15; C-5
 JDO (job descriptor ordinal) C-5
 Job 4-121; C-5
 Block C-5
 Category 3-31
 Controller 8-87
 Dayfile 3-17; C-5
 Processing example 3-18
 Resource limits 4-121
 Job descriptor number (JDN) 3-5,7,15; C-5
 Job descriptor ordinal (JDO) C-5
 Job descriptor table C-5
 Job termination 3-17
 Jobs waiting
 Interactive 3-8

K blocking 2-24
 K display 8-91
 KERNEL 1-4

L record format 2-21
 LABEL control statement 2-28; 4-55
 Label tape file 4-55
 Labeled common 4-72
 Labeling tapes F-1
 Labels
 Tapes 6-1
 Large page 1-2,4; C-5
 Last-group-file 2-7; C-5
 LB blocking 2-23
 LB tape format 2-35
 LCN (loosely coupled network) 1-3; 3-21;
 C-6
 Level number C-5
 Library 4-129; 5-8; C-5
 Library editing 4-94
 LID (logical identifier) C-5
 LIMITS control statement 4-60.1
 Linker 4-70
 Utility 3-12
 Linking object modules 4-69
 List
 Entry format 9-92
 File 5-9
 Pools 9-92
 User validations 4-60.1
 LISTAC control statement 4-61
 LISTAC output 4-63
 LOAD
 Control statement 4-74.1
 Execution 4-78.1
 Load map 4-66
 Utility 4-65
 LOADPF
 Control statement 4-80
 Example 4-87
 Output 4-86.1
 Local file 2-8; C-5
 Status 4-49
 Logging in to CYBER 200 OS 3-5
 Logging out of CYBER 200 OS 3-10
 Logical file name 2-28
 Logical identifier (LID) C-5
 Logical partitions 9-55,98
 Logical record format 2-18,33
 Logical record unit (LRU) 2-33; 9-17; C-6
 Format 9-17
 Logical structure conversions 3-26
 LOGIN command 3-5
 Look directive 6-13
 LOOK utility 6-13
 Loosely coupled network 1-3; 3-21; C-6
 LRU (logical record unit) 2-33; 9-17; C-6
 Format 9-17

Mainframe description 1-2
 Maintenance control unit (MCU) 1-2
 Map C-6
 Mapped files 2-17
 Mapping 3-2
 Parameter 4-72.1
 Mapping virtual space 9-70
 Mass storage C-6
 Characteristics 2-26
 Files 2-26; 9-70
 Mass storage files, copying 4-26
 Master account number 3-4
 Master control character 5-4
 Master project number C-6
 Master user C-6
 Matching substitution values to formal
 parameters 4-8
 Maximum file size 2-27
 Maximum physical record unit (MPRU)
 2-23,36
 Maximum validation 3-4
 MBKPT/MBKPTR Debug directive 6-10.1
 MCU (maintenance control unit) 1-2
 Memory overcommitment 3-15
 Message interrupts 8-35
 Message sending 8-82
 Message to dayfile 8-84
 Message to job 8-89
 Messages B-1,26
 System utility B-26
 MFGIVE control statement 3-25
 MFLINK control statement 4-88
 MFQUEUE control statement 3-23; 4-91
 MFTAKE control statement 3-25
 Minus page 2-2; C-6
 Miscellaneous table 8-12
 Mode of system 1-4
 Modify access permission 2-14
 Modmerge file 4-94
 Monitor mode 1-4
 MOVE Update directive 5-21
 MPRU (maximum physical record unit) 2-23,36
 Multifile name 2-28
 Multifile sets 2-31; 4-58; 9-67; C-6
 Multivolume tape files 2-29
 Multivolume tapes F-14
 Mutually exclusive SIL parameters 8-6

NAD (Network Access Device) 1-3
 Nesting of procedures 4-8
 Network 1-3
 Network Access Device (NAD) 1-3
 New program library 5-7
 No-operation keywords 8-7
 Non-I/O call 8-1
 Nonprivileged user C-6

Nonstandard labeled tape files 2-30
 Nonvariable (NV) tape format 2-36
 NORERUN control statement 4-93
 Norerun status 4-93
 NOS internal blocking 2-23
 NPSCALL 1-4
 Numbered common 4-72
 NV tape format 2-36

Object code 2-2
 File C-6
 Object module 4-69
 Octal to hexadecimal conversion A-5
 Old program library 5-8
 OLE control statement 4-94
 Omitting substitution values 4-10
 Online tapes 2-28; 4-84
 OP request line 3-9
 Opening a file 9-3,40
 Operating system 1-4
 Operator 8-90
 Operator message request 3-10
 Operator messages 4-119
 Optional SIL parameters 8-6
 Optional tape label formats F-8
 Optional tape labels 2-31
 Output file 2-4; C-6
 Error processing 2-7
 Routing 3-23
 Status 4-110
 OUTPUT file 3-19
 Output-file-family 2-7; C-7
 Overcommitment percentage 3-15
 Overlays 1-5
 Ownership 2-9; C-7
 Changing 4-53

P request line 3-8
 PACCESS control statement 4-97
 Pack file index (PFI) C-7
 Pack number C-7
 Page C-7
 Page fault C-7
 Page grouping 4-71
 PAGE Look directive 6-16
 Page sizes 1-2; 4-74
 Page table 1-6
 PAGER 1-4
 Paging 1-2,6
 Paired parameter format 8-6
 Parameter formats 4-4,9
 Parameter value ranges 9-9
 Partial logical partitions 9-57,100
 Partition C-7
 Delimiters 9-29
 PASSWORD control statement 4-98

PATTACH control statement 4-99
 PATTERN Look directive 6-16
 Patterning a file 2-9
 PCREATE control statement 4-100
 PDELETE control statement 4-101
 PDESTROY control statement 4-102
 PDETACH control statement 4-102
 Peripheral operating system 1-2
 Permanent file 2-8; C-7
 AUDIT request 3-25
 Information 4-15
 Requests 3-24
 Transfer 4-88
 Permanent File Transfer Facility
 (PTF) 3-21; C-7
 Permanent File Transfer Facility Servicer
 (PTFS) 3-21; C-7
 Permission 3-4
 Permissions listing 4-61
 PERMIT control statement 4-103
 PFI (pack file index) C-7
 PFILES control statement 4-105
 Physical address C-7
 Physical identifier (PID) C-7
 Physical record unit (PRU) 2-33; C-8
 Terminator 2-35
 PID (physical identifier) C-7
 Pool 2-12; C-7
 Boss 2-12; C-7
 Files 2-12; C-8
 List users 9-93
 Member 2-12; C-8
 Remove access 9-93
 Pool access 4-97
 Removing 4-101
 Pools listing information 4-105
 Positioning a file 4-128.1
 PR request line 3-8
 Preallocation C-8
 Preparing a file for I/O 9-3
 Presetting labeled common 4-73
 Print control characters 2-6
 Print files 2-5
 PRINT Look directive 6-16
 Priority 3-15
 Private files 2-12; C-8
 Privileged system tasks 1-4
 Privileged user 2-12; C-8
 PROC control statement 4-6.3
 Procedure 4-6.3
 Procedure file C-8
 PROCEED control statement 3-18; 4-48,106
 Processing 2-28; 3-16
 Production files 4-36.3,51,63,80,103; 6-2;
 8-22; 9-89,90
 Production user numbers 4-60.1; 6-2; 8-54
 Program breakpoint 6-2
 Program call 8-80

Program states	3-9	Q5GETCTS call	8-40
Project number	3-4; 4-21; C-8	Q5GETFIL call	9-40
Prompt	2-38	Q5GETFIT call	9-50
Prompting for parameter values	4-5	Q5GETIIP call	8-41
Propagation of deck changes	5-18	Q5GETIRF call	8-42
Provide information on segment locations	4-34.2	Q5GETLP call	8-43
PRU (physical record unit)	2-33; C-8	Q5GETMCE call	8-44
Terminator	2-35	Q5GETMCR call	8-45
Pseudo file	4-42	Q5GETMOP call	8-47
PTF (Permanent File Transfer Facility)	3-21; C-7	Q5GETMPG call	8-48
PTFS (Permanent File Transfer Facility Servicer)	3-21; C-7	Q5GETN call	9-55
Public files	2-13; C-8	Q5GETP call	9-57
PULLMOD directive	5-22	Q5GETPFI call	8-49
Pullmod file	5-9	Q5GETTL call	8-50
PURDECK directive	5-22.1	Q5GETTN call	8-51
PURGE control statement	4-106.1	Q5GETUID call	8-53
PURGE Update directive	5-23	Q5GIVE call	9-60
Purging files	4-106.1	Q5INIT call	8-55
Put a buffer record	9-97	Q5INITCH call	8-56
PXXfamnm file	2-5; 3-18	Q5JOBFILE file	3-16
		Q5LABEL call	2-31; 9-63
		Q5LFIHIR call	8-58
		Q5LFIPOL call	8-60
		Q5LFIPRI call	8-63
		Q5LFIPUB call	8-66
		Q5LSTBUT call	8-68
		Q5LSTCH call	8-69
Q statement	4-107	Q5LSTSTB call	8-71
QTF (Queue File Transfer Facility)	3-21	Q5LSTTCB call	8-72
QTFS (Queue File Transfer Facility Servicer)	3-6,21	Q5MAPIN call	9-70
Queue File Transfer Facility (QTF)	3-21	Q5MAPOUT call	9-70
Queue File Transfer Facility Servicer (QTFS)	3-6,21	Q5MEMORY call	8-73
Queue file transfers	3-22	Q5OPEN call	9-74
Quick update mode	5-7	Q5PATACH call	9-85
Q5ADVISE call	8-8	Q5PCREAT call	9-86
Q5ATTACH call	9-10	Q5PDESTR call	9-87
Q5CHANGE call	9-12	Q5PDTACH call	9-88
Q5CHECK call	9-16	Q5PERMIT call	9-89
Q5CHECKB call	9-19	Q5PGRACC call	9-91
Q5CLIOER call	9-21	Q5POOLS call	9-92
Q5CLOSE call	9-22	Q5PREACC call	9-93
Q5CPUTIM call	8-10	Q5PURGE call	9-94
Q5DAYFLE file	3-16	Q5PUSERL call	9-96
Q5DCDDST call	8-11	Q5PUTB call	9-97
Q5DCDMSC call	8-12	Q5PUTN call	9-98
Q5DCDPFI call	8-18	Q5PUTP call	9-100
Q5DCDPLB call	8-29	Q5READ call	9-102
Q5DEFINE call	9-25	Q5RECALL call	8-74
Q5DESBIF call	8-31	Q5REDUCE call	9-106
Q5DISAMI call	8-32	Q5REELSW call	9-107
Q5DISATI call	8-33	Q5REELSW routing	2-29
Q5DMPACT call	8-34	Q5REPREV call	8-75
Q5ENAMI call	8-35	Q5RETFIT call	9-109
Q5ENATI call	8-37	Q5RETURN call	9-110
Q5ENDPAR call	9-29	Q5REWIND call	9-111
Q5GENFIT call	9-31	Q5RFI call	8-77
Q5GETACT call	8-38	Q5ROUTE call	9-112
Q5GETB call	9-39	Q5RQUEST call	9-115

Q5RUNBIF call 8-78
 Q5SETFIT call 9-122
 Q5SETLP call 8-79
 Q5SKIP call 9-126
 Q5SNDMCE call 8-80
 Q5SNDMCR call 8-82
 Q5SNDMDF call 8-84
 Q5SNDMJC call 8-87
 Q5SNDMJS call 8-89
 Q5SNDMOP call 8-90
 Q5SNDSTR call 8-92
 Q5TERM call 8-93
 Q5TERMCE call 8-94
 Q5TIME call 8-95
 Q5VRACC call 8-96
 Q5WRITE call 9-130
 Q9CI32 routine E-3
 Q9CI64 routine E-3
 Q9CLF routine E-7
 Q9CLI routine E-7
 Q9IC32 routine E-2
 Q9IC64 routine E-1
 Q9LCF routine E-7
 Q9LCI routine E-6
 Q9SPRINT 11-16
 Q9SCR 11-1
 Definitions 11-1
 Calling Conventions 11-3
 Descriptions 11-7
 Limits 11-15
 Q9xxx routine format E-6

 R record format 2-19
 Random access file transfers 3-25
 Read access permission 2-14
 READ Update directive 5-24
 Reading date 9-5
 Record C-8
 Count blocking 2-24
 Format 2-18
 Mark character 2-19
 Mark delimited record format 2-19
 Number 2-25
 Recording densities 2-32
 Records per block (RPB) 2-24
 Reducing file length 9-106
 Register file 1-6
 Releasing mass storage space 9-70
 Reloading files 4-80
 Remote host C-8
 Remote Host Facility (RHF) 1-5; 3-21;
 C-8
 Application 2-38
 Reloading 4-83
 Conversions 3-26
 Permanent file requests 3-24
 Structure conversion 4-90
 Remote system 3-6
 Removing user access to pool 4-101

Repositioning a sequential access
 file 9-126
 Reprieve subroutine 3-28
 REQUEST control statement 4-112
 ? request line 3-8
 Request lines 3-7
 REQUEST options 4-119
 Requesting and opening a file 9-40
 Requesting files 3-24
 Required ANSI tape label formats F-5
 Required SIL parameters 8-6
 RERUN control statement 4-120
 Rerun status 4-120
 Reserved file names 4-36.2
 Resident system 1-4
 Resource allocation 3-31
 RESOURCE control statement 3-22; 4-121
 Resource limits 4-134
 Resource usage statistics 4-134
 Resource validation 3-33
 Resources used 4-134
 Restarting a checkpointed file 7-5
 Restarting a task 4-120
 At security-sensitive sites 2-3
 From drop file 2-3
 From terminal 3-7,10
 RESTORE Debug directive 6-9
 Retrieving FIT field values 9-50
 Return code B-1
 RETURN control statement 4-124
 Return parameters 8-7
 Returning file 9-110
 REWIND control statement 4-126
 Rewinding files 9-111
 Rewinding tape 4-126
 RHF (Remote Host Facility) 1-5; 3-21; C-8
 Application 2-38
 Reloading 4-84
 Conversions 3-26
 Permanent file requests 3-24
 Structure conversion 4-90
 RMS (rotating mass storage) C-8
 ROLL Debug directive 6-6
 ROLL Look directive 6-16
 Rotating mass storage (RMS) C-8
 Route QTFS directive 3-23
 Routing
 Explicit file 4-91
 Routing files 4-132
 RPB (records per block) 2-24

 S request line 3-8
 Satisfying externals 4-69
 Save table 8-84
 Saving the drop file 2-4
 SBU (system billing unit) C-9
 SBU/STU accumulators 4-21
 Scalar C-8
 Scalar processor 1-2

Scheduling 3-15
 Scratch files 2-8; C-8
 Search hierarchy 2-12
 SEARCH Look directive 6-16
 Second time limit disable 3-30
 Security administrator 4-36.3,80;
 9-14,89,90
 Security levels 2-9; C-9
 Security-sensitive sites 2-3,9; 3-1,31; 6-2
 Segment(s) 2-27; C-9
 Location 4-34.2
 Selection number 3-15
 SEQ Look directive 6-16
 Sequence number 5-1
 Sequential access file organization 2-25
 Session 3-7
 Session termination request 3-10.1
 SET control statement 4-127
 Sharable files 2-16
 Shared SYSLIB C-9
 Shared utility C-9
 Short PRU 2-35; C-9
 SHRLIB (system shared library) 3-12;
 4-70,129; C-10
 SI tape format 2-35
 SIL (system interface language) 1-5; 8-1,6;
 9-1; C-9
 Single volume tapes F-13
 SKIP control statement 4-128.1
 Skipping tape 4-128.1
 SLGEN control statement 4-129
 SLGEN example 4-131
 Small page C-9
 Small page size 1-2
 SNAP Debug directive 6-11
 Source decks 5-1
 Source file C-9
 Source file maintenance utility 5-1
 Space allocation 2-17
 Space initialization 4-73
 \$ special character 3-7
 Specification 3-15,16; 4-119
 Specification of execution 3-10
 Specification on statements 4-4
 Standalone SIL parameter format 8-6
 Starting a previously initialized
 controllee 8-92
 STAT Debug directive 6-12
 Static execution 3-12
 Statistics buffer 8-71
 Status 4-107; 9-16
 Status code categories 8-5
 Status listing 4-107
 STEP Debug directive 6-11
 Stepping through a program 6-2
 Storage in controllee file 4-69
 Structure 3-14
 STU (system time unit) C-10
 SU request line 3-8
 SUBMIT control statement 4-132
 Submitting a job file 4-91
 Submitting file to a queue 4-132
 Subroutine 8-77
 Subroutine traceback
 DUMP 6-23
 Substitution value 4-9
 SUMMARY control statement 4-134
 Suppressing parameter substitution 4-11
 Suspending task execution 8-74
 SWITCH control statement 4-136
 SYSLIB file 3-20
 System 1-4
 Access 3-4
 Billing unit (SBU) C-9
 Block record format 2-22
 Dayfile C-9
 Description 1-1
 Error codes B-104
 Interface language (SIL) 1-5; 8-1,6;
 9-1; C-9
 Message Call parameters 8-5; 9-9; C-9
 Pool 2-13; C-9
 Tasks 1-6
 Time unit (STU) C-10
 User C-10
 User number 3-4
 Utility error B-26
 System error code table B-104
 System-generated drop file 2-2
 System shared library (SHRLIB) 3-12;
 4-70,129; C-10
 T request line 3-8
 Tape 2-37
 Advanced System (ATS) 1-3
 Count discrepancy 2-31
 Data
 Blocking 2-23
 Conversion 2-32
 Organization 2-23,33
 Recording 2-23,32
 Density 2-32
 Drive reservation 2-28
 Error
 Processing 2-36
 Recovery 2-36
 Summary 2-36
 File request 4-118
 Files 2-28; 9-48
 Formats 2-23,34
 LB 2-35
 SI 2-35
 V 2-36
 I/O requests 9-31
 Label processing 9-29
 Labeling 2-29
 Labels and formats 6-1
 Operator communication 2-29

Partition delimiters 9-30
 Processing 9-29
 Repositioning a file 4-126
 Request 2-28
 Resource parameter 2-28
 Rewinding 4-126
 Volume 2-29
 Tape blocks per group F-11
 Tape drive reservation 2-28.1; 4-123
 Tape error codes B-106
 Tape files 2-28.1; 9-110.1
 Copying 4-26
 Tape group separators F-10
 Tape I/O requests 9-17
 Tape labels and formats F-1
 Target page size 4-74
 Task 1-6; 3-16; C-10
 Task interrupt request 3-10
 Task resource limits 3-10
 TASKATT control statement 4-139
 TEMNEWPL file 5-6
 Terminal message interrupts 3-7
 Terminating a task and its controllees 8-93
 Termination 3-10.1,17,28
 Termination value 3-17; 8-93; B-1
 Threshold value 3-16; 4-140; B-1; C-10
 Time available 3-8
 Time consumed 3-8
 Timecard buffer 8-72
 Transfer conversions 3-26
 Transferring permanent files 4-88
 Transferring random access files 3-25
 Transfers between systems 3-21
 Trunk lines 1-3
 TSP parameter 4-73
 TV control statement 4-140

 U record format 2-19
 UEP (user error processing) 2-37
 Undefined record format 2-19
 Unlabeled tape files 2-30; F-4
 UPC (user project control) 4-21; C-10
 Update 5-1
 UPDATE control statement 5-26.1
 / Update directive 5-25
 UPDATE processing 5-4
 Usage 3-10,28
 Use of system 3-2
 User
 Directory changes 3-4
 Error Processing (UEP) 2-37
 Label processing 2-29
 Number C-10
 Number 000000 2-13
 Permission 3-4
 Reprive 3-28
 Validation information 4-142
 USER control statement 3-6,22.1; 4-142
 User-generated drop file 2-3

 User project control (UPC) 4-21; C-10
 User validations 4-60.1
 USER1 C-10
 User-1 C-10
 Users with pool access 2-12
 Using interactive terminal files 2-37
 Using SIL calls 9-1,9

 V tape format 2-36
 Validation 3-4; 4-60.1
 Validation of user 3-4
 Variable rate accounting 4-47
 Variable rate table 4-47
 Vector C-10
 Vector processor 1-2
 Virtual
 Address C-10
 Code file C-10
 Files 2-2
 Memory 1-2; C-10
 Space mapping 3-2
 System tasks 1-4
 VIRTUAL Look directive 6-16
 Virtual memory addressing 1-6
 Virtual space requirements 8-8
 Volume 2-29; C-10
 Assignment 2-29
 Serial number 2-29
 Switching 2-29
 VOL1 label 2-30
 VRI 4-47
 VSN list 2-29
 VSOS introduction 1-2

 W record format 2-20
 Warning SIL errors 8-5
 WIDTH Update directive 5-24
 Word C-10
 WORD Look directive 6-16
 Word size 1-5
 Working set C-10
 Working set size limit 4-121
 Workstation 8-17,22; 9-12
 Write access permission 2-14
 Write additional tape volume labels
 9-107,132
 Write label processing 2-29
 Writing data 9-5

 XIOCALL 1-4

 YANK Update directive 5-24
 YANKDECK Update directive 5-25

 Zero-length PRU C-10

 60-bit conversion 2-32
 7639 disk controller 1-2
 819 disk unit 1-2,3
 * control statement 4-22

Please fold on dotted line;
seal edges with tape only.

FOLD

FOLD

FOLD



BUSINESS REPLY MAIL

First-Class Mail Permit No. 8241 Minneapolis, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CONTROL DATA

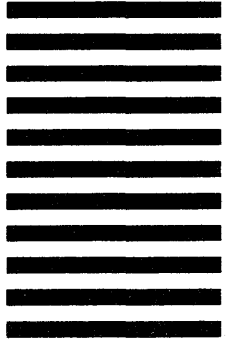
Technical Publications

ARH219

4201 N. Lexington Avenue

Arden Hills, MN 55126-9983

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



COMMENT SHEET

MANUAL TITLE: CDC VSOS Version 2 Reference Manual, Volume 1 of 2

PUBLICATION NO.: 60459410

REVISION: J

NAME: _____

COMPANY: _____

STREET ADDRESS: _____

CITY: _____ STATE: _____ ZIP CODE: _____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

Please Reply

No Reply Necessary

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

FOLD ON DOTTED LINES AND TAPE

SIL CALL INDEX

This index lists each system interface language (SIL) call and the page on which it is described.

Q5ADVISE	8-8	Q5GETMPG	8-48	Q5PUSERL	9-96
Q5ATTACH	9-10	Q5GETN	9-55	Q5PUTB	9-97
Q5CHANGE	9-12	Q5GETP	9-57	Q5PUTN	9-98
Q5CHECK	9-16	Q5GETPFI	8-49	Q5PUTP	9-100
Q5CHECKB	9-19	Q5GETTL	8-50	Q5READ	9-102
Q5CLIOER	9-21	Q5GETTN	8-51	Q5RECALL	8-74
Q5CLOSE	9-22	Q5GETUID	8-53	Q5REDUCE	9-106
Q5CPUTIM	8-10	Q5GIVE	9-60	Q5REELSW	9-107
Q5DCDDST	8-11	Q5INIT	8-55	Q5REPREV	8-75
Q5DCDMSC	8-13	Q5INITCH	8-56	Q5RETFIT	9-109
Q5DCDPFI	8-18	Q5LABEL	9-63	Q5RETURN	9-110
Q5DCDPLB	8-29	Q5LFHIR	8-58	Q5REWIND	9-111
Q5DEFINE	9-25	Q5LFIPOL	8-60	Q5RFI	8-77
Q5DESBIF	8-31	Q5LFIPRI	8-63	Q5ROUTE	9-112
Q5DISAMI	8-32	Q5LFIPUB	8-66	Q5RQUEST	9-115
Q5DISATI	8-33	Q5LSTBUT	8-68	Q5RUNBIF	8-78
Q5DMPACT	8-34	Q5LSTCH	8-69	Q5SETFIT	9-122
Q5ENAMI	8-35	Q5LSTSTB	8-71	Q5SETLP	8-79
Q5ENATI	8-37	Q5LSTTCB	8-72	Q5SKIP	9-126
Q5ENDPAR	9-29	Q5MAPIN	9-70	Q5SNDMCE	8-80
Q5GENFIT	9-31	Q5MAPOUT	9-72	Q5SNDMCR	8-82
Q5GETACT	8-38	Q5MEMORY	8-73	Q5SNDMDF	8-84
Q5GETB	9-39	Q5OPEN	9-74	Q5SNDMJC	8-87
Q5GETCTS	8-40	Q5PATAch	9-85	Q5SNDMJS	8-89
Q5GETFIL	9-40	Q5PCREAT	9-86	Q5SNDMOP	8-90
Q5GETFIT	9-50	Q5PDESTR	9-87	Q5SNDSTR	8-92
Q5GETIIP	8-41	Q5PDTACH	9-88	Q5TERM	8-93
Q5GETIRF	8-42	Q5PERMIT	9-89	Q5TERMCE	8-94
Q5GETLP	8-43	Q5PGRACC	9-91	Q5TIME	8-95
Q5GETMCE	8-44	Q5POOLS	9-92	Q5VRACC	8-96
Q5GETMCR	8-45	Q5PREACC	9-93	Q5WRITE	9-130
Q5GETMOP	8-47	Q5PURGE	9-94		

