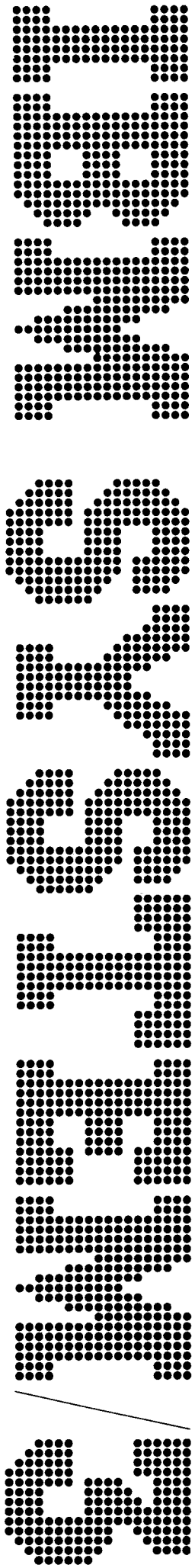


IBM System/3 Model 15 System Control Programming Concepts and Reference Manual

Program Number 5704-SC2

GC21-5162-1
File No. S3-36

Preface

This manual provides programmers with the information needed to run programs on the IBM System/3 Model 15 and to use the system service programs for doing jobs such as preparing disks for use or updating system libraries. See *How To Use This Manual* for additional information.

Related Publications

- *IBM System/3 Model 15 Introduction,* GC21-5094
- *IBM System/3 Disk Concepts and Planning Guide,* GC21-7571
- *IBM System/3 Model 15 Operator's Guide,* GC21-5075
- *IBM System/3 Model 15 System Generation Reference Manual,* GC21-7616
- *IBM System/3 Model 15 System Messages,* GC21-5076
- *IBM System/3 Communications Control Program Messages Manual,* GC21-5170
- *IBM System/3 Communications Control Program System Reference Manual,* GC21-7620
- *IBM System/3 3741 Reference Manual,* GC21-5113
- *IBM System/3 Model 15 User's Guide to Spooling,* GC21-7632
- *IBM System/3 Model 15 System Control Programming Macros Reference Manual,* GC21-7608
- *IBM System/3 Overlay Linkage Editor Reference Manual,* GC21-7561
- *IBM System/3 Multiline/Multipoint Binary Synchronous Communications Reference Manual,* GC21-7573

Note: Information about the system control program (Program Number 5704-SC1) is in the *IBM System/3 Model 15 System Control Programming Reference Manual,* GC21-5077.

Second Edition (September 1978)

This is a major revision of, and obsoletes, GC21-5162-0 and technical newsletter GN21-5550. Because the changes and additions are extensive, this publication should be reviewed in its entirety.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, refer to the latest *IBM System/3 Bibliography,* GC20-8080 for the editions that are applicable and current.

Use this publication only for the purposes stated in the *Preface.*

Publications are not stocked at the address below. Requests for copies of IBM publications and for technical information about the system should be made to your IBM representative or to the branch office serving your locality.

This publication could contain technical inaccuracies or typographical errors. Use the Reader's Comment Form at the back of this publication to make comments about this publication. If the form has been removed, address your comments to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Contents

HOW TO USE THIS MANUAL ix

MODEL 15D INTRODUCTION xi

PART 1. OCL STATEMENTS 1-1

INTRODUCTION TO OCL STATEMENTS 1-3
What is OCL? 1-3
OCL and the Job Stream 1-3
Organization of Part 1 1-4
Coding Rules 1-4
Types of Information 1-4
General Coding Rules 1-4
Statement Length 1-6
STATEMENT DESCRIPTIONS 1-8
ASSIGN STATEMENT 1-21
BSCA STATEMENT 1-23
CALL STATEMENT 1-24
COMPILE STATEMENT 1-25
DATE STATEMENT 1-27
FILE STATEMENT (SINGLE VOLUME DISK FILES) . 1-29
FILE STATEMENT (MULTIVOLUME DISK FILES) . 1-40
FILE STATEMENT (SINGLE VOLUME TAPE FILES) 1-45
7-Track Considerations 1-52
Tape File Statement Summary 1-53
Combinations of 7-Track Specifications 1-54
FILE STATEMENT (MULTIVOLUME TAPE FILES) . 1-55
FILE STATEMENT (MULTIPLE TAPE VOLUMES) . 1-57
Prepositioned Tapes (SEQNUM-X on the FILE Statement) 1-58
Restrictions on the Use of Multifile Tapes . . . 1-58
Standard Labeled Files 1-59
Nonstandard Labeled Files 1-61
Unlabeled Volumes 1-61
REEL Parameter on FILE Statement 1-62
FILE STATEMENT (DEVICE INDEPENDENT FILES) 1-63
HALT STATEMENT 1-65
IMAGE STATEMENT 1-66
Characters from the System Input Device . . . 1-67
Characters from the Source Library 1-67
INCLUDE STATEMENT 1-69
JOB STATEMENT 1-71
LOAD AND LOAD * STATEMENT 1-75
LOG STATEMENT 1-79
NOHALT STATEMENT 1-82
PAUSE STATEMENT 1-84
PRINTER STATEMENT 1-85
PUNCH STATEMENT 1-88
READER STATEMENT 1-91
RUN STATEMENT 1-92
SWITCH STATEMENT 1-93
/& STATEMENT 1-95
/. STATEMENT 1-96
*(COMMENT) STATEMENTS 1-99
/* STATEMENT 1-100

PART 2. SYSTEM CONCEPTS AND FACILITIES . . . 2-1

PROGRAM FACILITIES 2-3
System/3 Model 15 Programming Support 2-3
Program Concepts 2-4
Source Programs 2-4
Object Modules 2-4
Load Modules 2-5
Program and Partition Sizes 2-8
Greater Than 48K Programs 2-9
External Buffers 2-9
FILE FACILITIES 2-11
File Definition 2-11
File Organization 2-11
File Processing 2-11
File Creation 2-12
File Location 2-13
Automatic File Allocation 2-13
File Services 2-15
Scheduler Work Area 2-15
File Sharing 2-16
Compatible Access Methods for File Sharing . . . 2-17
DTF (Define the File) 2-17
SDTF (Share Define the File) 2-17
FSQE (File Share Queue Element) 2-18.1
File Share Area 2-18.1
Doubly-Defined Files 2-18.1
Considerations and Restrictions 2-18.2
General Results When the 2 or 3 Option is Selected for a Message 2-19
Work Files 2-20
Main Storage Requirements 2-20
RPG II 2-21
COBOL 2-21
FORTRAN 2-21
CCP/Disk Sort 2-22
Basic Assembler 2-22
Overlay Linkage Editor 2-23
Large Index Files 2-23
Multivolume Disk Files 2-24
Multivolume Tape Files 2-25
Multifile Tape Volumes 2-25
Null Files on Tape 2-25
Programming Considerations 2-26
LIBRARY FACILITIES 2-26.1
Library Definition 2-26.1
Source Library 2-26.1
Object Library 2-27
Library Locations 2-29
Storing Programs 2-30
Sample Statements 2-31
Procedures 2-32
Example 2-33
Nested Procedures 2-35
Cataloging to an Active Library 2-39
User Considerations 2-41
SYSTEM FACILITIES 2-42
Initial Program Load (IPL) 2-42
Program Execution 2-43
Job and Step Processing 2-44

External Indicators . . . 2-48
System Severity Codes . . . 2-48
Job Stream Example . . . 2-48
Multiprogramming . . . 2-52
Operating in a Multiprogramming Environment . . . 2-53
Multiprogramming Considerations and Restrictions . . . 2-57
Sharing Access to Added Records . . . 2-59
Multiprogramming Examples . . . 2-60
Date Support . . . 2-63
System Date . . . 2-63
Partition Date . . . 2-63
Interval Timer . . . 2-63
System Input Device . . . 2-64
System Log Device . . . 2-65
System Print Device . . . 2-66
System Punch Device . . . 2-66
System History Area . . . 2-66
Spooling . . . 2-67
Checkpoint/Restart . . . 2-68
Inquiry Program . . . 2-69
System Integrity . . . 2-69
Automatic Message Restart (Unit Record Restart) . . . 2-70
Unit Record Restart (System Generation Option) . . . 2-70
Extended Restart (System Generation Option) . . . 2-70
Main Storage Usage . . . 2-71

PART 3. DISK STORAGE . . . 3-1

DIRECT ACCESS STORAGE . . . 3-3
3340 Direct Access Storage Facility . . . 3-3
3344 Direct Access Storage . . . 3-4
Simulation Areas . . . 3-6
Interchanging Data Modules (3340) . . . 3-6
Accessing Simulation Areas . . . 3-7
Assigning Simulation Areas . . . 3-7
Simulation Area Reassignment . . . 3-9
Number of Simulation Area Assignments . . . 3-9
Main Data Areas . . . 3-10
Disk Space Allocation . . . 3-10
Considerations and Restrictions . . . 3-10
Alternate Tracks . . . 3-11
File Processing Considerations . . . 3-11
Cylinder 0 Format . . . 3-12
Considerations and Restrictions . . . 3-13
Initial Program Load (IPL) . . . 3-13

PART 4. SYSTEM SERVICE PROGRAMS . . . 4-1

INTRODUCTION . . . 4-3
Programming Considerations . . . 4-4
Control Statements . . . 4-4
Writing Control Statements for System Service Programs . . . 4-4
Coding Rules . . . 4-5
END Control Statement . . . 4-5
Placement of Control Statements in the Job Stream . . . 4-5
Special Meaning of Capital Letters, Numbers, and Special Characters . . . 4-5
Device Codes . . . 4-6

ALTERNATE TRACK ASSIGNMENT PROGRAM—$ALT . . . 4-7
Program Description . . . 4-7
Control Statement Summary . . . 4-7
Parameter Summary . . . 4-8
Parameter Descriptions . . . 4-8
PACK Parameter . . . 4-8
UNIT Parameter . . . 4-8
VERIFY Parameter . . . 4-8
ASSIGN Parameter . . . 4-8
Unconditional Assignment . . . 4-9
Conditional Assignment . . . 4-9
OCL Considerations . . . 4-10
Examples . . . 4-10
Conditional Assignment . . . 4-10
Unconditional Assignment . . . 4-10
Messages for Alternate Track Assignment . . . 4-11
ALTERNATE TRACK REBUILD PROGRAM—$BUILD . . . 4-12
Program Description . . . 4-12
Control Statement Summary . . . 4-12
Parameter and Substitute Data Summary . . . 4-13
Parameter and Substitute Data Descriptions . . . 4-14
PACK Parameter . . . 4-14
UNIT Parameter . . . 4-14
TRACK Parameter . . . 4-14
LENGTH Parameter . . . 4-14
DISP (Displacement) Parameter . . . 4-14
Substitute Data . . . 4-14
OCL Considerations . . . 4-14
Examples . . . 4-15
CONFIGURATION RECORD PROGRAM—$CNFIG . . . 4-15
Program Description . . . 4-17
Changing the Configuration Record . . . 4-17
Control Statement Summary . . . 4-17
Parameter Summary . . . 4-18
Parameter Descriptions . . . 4-20
AUTHORIZE Parameter (QCOPY) . . . 4-22
CARD Parameter (DEFCN) . . . 4-22
Code-Area Parameter (ASNPx) . . . 4-22
CYL Parameter (SPCYL) . . . 4-22
D Parameter (TSTAMP) . . . 4-22
DATE Parameter (FORMAT) . . . 4-22
DEVICE Parameter (LOGPx, SYINx, SYPCx, SYPRx) . . . 4-22.1
EJECT or NOEJECT Parameter (LOGPx) . . . 4-22.1
ERASE Parameter (QCOPY) . . . 4-22.1
EXTENDED Parameter (READY) . . . 4-22.1
FORM Parameter (DEFFN) . . . 4-22.1
FS Parameter (SIZE) . . . 4-22.1
HALT Parameter (HLTPx) . . . 4-22.1
HALT Parameter (SHA) . . . 4-22.1
I Parameter (TSTAMP) . . . 4-22.2
IDELETE Parameter (ITYPE) . . . 4-22.2
M Parameter (SPOPT) . . . 4-22.2
OCL Parameter (BLANK) . . . 4-22.2
PACK Parameter (CATLG) . . . 4-22.2
PRINT Parameter (AUTST, AUTWT) . . . 4-22.2
PUNCH Parameter (AUTST, AUTWT) . . . 4-22.2
P1 Parameter (SIZE) . . . 4-23
P2 Parameter (SIZE) . . . 4-23

P3 Parameter (SIZE) 4-23
RATIO Parameter (FSHARE) 4-24
READ Parameter (AUTST) 4-24
RETAIN Parameter (MESSAG) 4-24
RQPTY Parameter (QCOPY) 4-24
SEQUENCE Parameter (PRIORITY) 4-24
SHARE Parameter (CONSOL) 4-24.1
SHARE Parameter (FSHARE) 4-24.1
SYS Parameter (SIZE) 4-24.1
TRACKS Parameter (SHA) 4-24.1
TRACKS Parameter (SPEXT) 4-24.1
UNIT Parameter (SPDSK) 4-24.1
Considerations and Restrictions 4-24.1
OCL Considerations 4-24.2
Examples 4-25
COPY/DUMP PROGRAM—$COPY 4-27
Program Description 4-27
COPYPACK 4-27
COPYFILE 4-27
Control Statement Summary 4-29
Parameter Summary 4-31
Parameter Descriptions 4-34
FROM and TO Parameters (COPYPACK) . . . 4-34
PACKIN and PACKO Parameters (COPYPACK) 4-34
COPYPACK Considerations 4-34
OUTPUT Parameter (COPYFILE) 4-34
DELETE Parameter (COPYFILE) 4-36
REORG (Reorganize) Parameter (COPYFILE) . . 4-36
LENGTH Parameter (COPYFILE) 4-36
KEY and PKY Parameters (SELECT) 4-36
RECORD Parameters (SELECT) 4-37
FILE Parameter (SELECT) 4-37
LENGTH and LOCATION Parameters (KEY) . . 4-37
DATAMGMT Parameter (OUTDM) 4-37
FROM Parameter (ACCESS) 4-37
CYLINDER Parameter (ACCESS) 4-37
SECTOR Parameter (ACCESS) 4-37
TRACK Parameter (ACCESS) 4-38
RECL Parameter (ACCESS) 4-38
DISP Parameter (ACCESS) 4-38
Copying Multivolume Files 4-38
Maintaining Proper Volume Sequence Numbers 4-38
Maintaining Correct Relative Record Numbers 4-38
Direct File Attributes 4-38
Copying Multivolume Indexed Files 4-38
Tape File Considerations 4-39
Diskette File Considerations 4-39
Card Input Considerations 4-40
Card Output Considerations 4-41
File Recovery Considerations 4-41
OCL Considerations 4-41
Examples 4-42
DUMP/RESTORE PROGRAM—$DCOPY 4-62
Program Description 4-62
Control Statement Summary 4-62
Parameter Summary 4-63
Parameter Descriptions 4-63
FROM and TO Parameters (COPYPACK) . . . 4-63
PACK Parameter (COPYPACK) 4-64
SYSTEM Parameter (COPYPACK) 4-64
BACKUP Parameter (COPYPACK) 4-64
Dump/Restore Considerations 4-65
OCL Considerations 4-65
FILE Statement Considerations 4-65
Messages for DUMP/RESTORE 4-66
Examples 4-66
FILE Statement: From Disk to Tape 4-67
Control Statements 4-67
FILE Statement: From Tape to Disk 4-68
Control Statement: From Disk to Diskette . . . 4-69
Control Statement: From Disk to Tape 4-70
Programming Considerations 4-70
FILE DELETE PROGRAM—$DELET 4-71
Program Description 4-71
Deleting Files 4-71
Freeing Space on an Area 4-71
Control Statement Summary 4-72
Parameter Summary 4-73
Parameter Descriptions 4-74
PACK Parameter 4-74
UNIT Parameter 4-74
LABEL Parameter 4-74
DATE Parameter 4-74
DATA Parameter 4-74
OCL Considerations 4-75
Examples 4-75
Deleting One of Several Files Having the Same Name 4-75
Freeing Allocated But Unused Space on an Area . 4-77
FILE COMPRESS PROGRAM—$FCOMP 4-78
Program Description 4-78
Move and Copy Functions 4-78
Move Function 4-78
Copy Function 4-78
Backup and Restore Functions 4-79
Backup Function 4-79
Restore Function 4-79
Control Statement Summary 4-80
Parameter Summary 4-80
Parameter Description 4-82
FROM and TO Parameters (COPYFILE) 4-82
PACKIN and PACKO Parameters (COPYFILES) . 4-82
COMPRESS Parameter (COPYFILES) 4-82
FROM Parameter (TAPEFILES) 4-82
TO Parameter (TAPEFILES) 4-82
LABEL Parameter (TAPEFILES) 4-82
PACK Parameter (TAPEFILES) 4-82
SEQNUM Parameters (TAPEFILES) 4-82
COMPRESS Parameter (TAPEFILES) 4-83
Considerations and Restrictions 4-83
File Statement Considerations and Restrictions (Backup and Restore) 4-83
OCL Considerations 4-84
Examples 4-84
SYSTEM HISTORY AREA DISPLAY PROGRAM—$HIST 4-93
Program Description 4-93
Control Statement Summary 4-94
Parameter Summary 4-94
Parameter Descriptions 4-94
HISTORY Parameter 4-94
OCL Considerations 4-94
Examples 4-97
DISK INITIALIZATION PROGRAM—$INIT . . . 4-100
Program Description 4-100
Types of Initialization 4-100
Control Statement Summary 4-101

Parameter Summary 4-102
Parameter Descriptions 4-103
TYPE Parameter (UIN) 4-103
UNIT Parameter 4-104
VERIFY Parameter 4-104
ERASE Parameter (UIN) 4-104
Surface Analysis 4-104
PACK Parameter (VOL) 4-105
ID (Identification) Parameter (VOL) 4-105
NAME360 Parameter (VOL) 4-105
OLDPACK Parameter (VOL) 4-105
OCL Considerations 4-106
Examples 4-106
Primary Initialization of Two Volumes 4-106
Messages for Disk Initialization 4-107
CHAIN CLEANING PROGRAM—$KLEAN 4-108
Program Description 4-108
OCL Considerations 4-108
FILE AND VOLUME LABEL DISPLAY PROGRAM—$LABEL 4-109
Program Description 4-109
Storage Requirements 4-109
Control Statement Summary 4-110
Parameter Summary (Display Statement) 4-110
Parameter Descriptions 4-111
UNIT Parameter 4-111
LABEL Parameter 4-111
SORT Parameter 4-111
Entire Contents of VTOC 4-111
Meaning of VTOC Information 4-113
File Information Only 4-115
Example 4-116
Printing VTOC Information for Two Files 4-116
LIBRARY MAINTENANCE PROGRAM—$MAINT 4-117
Program Descriptions 4-117
Use of Disk Space 4-117
Organization of This Section 4-117
$MAINT—ALLOCATE FUNCTION 4-118
Uses 4-118
Control Statement Summary 4-118
Considerations and Restrictions 4-119
Parameter Summary 4-120
Parameter Descriptions 4-121
TO Parameter 4-121
SOURCE and OBJECT Parameters 4-121
DIRSIZE Parameter 4-121
SYSTEM Parameter 4-121
HISTORY Parameter 4-122
WORK Parameter 4-122
PACKO Parameter 4-123
Using the Allocate Function 4-124
Creating a Source Library (SOURCE-number) 4-124
Changing the Size of (Reallocating) a Source Library (SOURCE-number) 4-124
Deleting a Source Library (SOURCE-0) 4-125
Reorganizing a Source Library (SOURCE-R) 4-126
Creating an Object Library (OBJECT-number) 4-126
Changing the Size of (Reallocating) an Object Library (OBJECT-number) 4-127
Deleting an Object Library (OBJECT-0) 4-127
Reorganizing an Object Library (OBJECT-R) 4-127
Compress in Place (OBJECT-{R / number}) 4-127
$MAINT—COPY FUNCTION 4-128
Uses 4-128
Control Statement Summary 4-129
Reader-to-Library 4-129
File-to-Library 4-129
Library-to-File 4-129
Library-to-Library 4-130
Library-to-Printer-and/or-Card 4-131
Considerations and Restrictions 4-131
Parameter Summary 4-132
Library Directories 4-135
Source and Object Library Directories 4-135
System Directory 4-135
Naming Library Entries 4-135
Characters to Use 4-135
Length of Name 4-135
Restricted Names 4-135
Entries with the Same Name 4-135
Retain Types 4-136
Temporary Entries 4-136
Permanent Entries 4-136
Simulation Area Verification 4-136
Using the Copy Function 4-137
Reader-to-Library 4-137
File-to-Library 4-137
Library-to-File 4-138
Library-to-Library 4-138
Library-to-Printer and/or Card 4-139
$MAINT—DELETE FUNCTION 4-146
Uses 4-146
Control Statement Summary 4-146
Considerations and Restrictions 4-147
Parameter Summary 4-148
$MAINT—MODIFY FUNCTION 4-149
Uses 4-149
Considerations and Restrictions 4-149
Control Statement Summary 4-150
Parameter Summary 4-151
Remove, Replace, Insert Parameters 4-152
$MAINT—RENAME FUNCTION 4-153
Uses 4-153
Considerations and Restrictions 4-153
Control Statement Summary 4-153
Parameter Summary 4-154
OCL Considerations 4-155
$MAINT—EXAMPLES 4-156
SPOOL FILE COPY PROGRAM—$QCOPY 4-167
Program Description 4-167
Control Statement Summary 4-168
Parameter Summary 4-171
Parameter Descriptions—COPYSP 4-176
FROM and TO Parameters 4-176
Parameter Descriptions—COPYPRTQ 4-176
UNIT Parameter 4-176
FORMSNO Parameter 4-176
JOBN Parameter 4-176
STEPN Parameter 4-176
LENGTH Parameter 4-176
REMOVE Parameter 4-176
OUTPUT Parameter 4-176
FILE Parameter 4-177
HEADER Parameter 4-177
STOP Parameter 4-177

Parameter Descriptions—COPYPCHQ 4-177
UNIT Parameter 4-177
CARDNO Parameter 4-177
JOBN Parameter 4-177
STEPN Parameter 4-177
REMOVE Parameter 4-177
OUTPUT Parameter 4-178
FILE Parameter 4-178
HEADER Parameter 4-178
STOP Parameter 4-178
Parameter Descriptions—COPYRDRQ 4-178
UNIT Parameter 4-178
RECL Parameter 4-178
INPUT Parameter 4-178
FILE Parameter 4-179
KEY Parameter 4-179
LOKEY Parameter 4-179
HIKEY Parameter 4-179
LOREC Parameter 4-180
HIREC Parameter 4-180
OUTPUT Parameter 4-180
JOBN Parameter 4-180
PARTITION Parameter 4-180
REMOVE Parameter 4-180
Parameter Descriptions—COPYCTRL 4-181
FILE Parameter 4-181
Parameter Descriptions—DISPLAY 4-181
UNIT Parameter 4-181
OUTPUT Parameter 4-181
FILE Parameter 4-181
Q Parameter 4-181
Parameter Descriptions—RESTORE 4-182
FILE Parameter 4-182
JOBN Parameter 4-182
STEPN Parameter 4-182
FORMSNO Parameter 4-182
CARDNO Parameter 4-182
STOP Parameter 4-182
OUTPUT Parameter 4-182
UNIT Parameter 4-182
Parameter Descriptions—COPYQ 4-183
Q Parameter 4-183
FROM Parameter 4-183
TO Parameter 4-183
JOBN Parameter 4-183
STEPN Parameter 4-183
REMOVE Parameter 4-183
PARTITION Parameter 4-183
PRIORITY Parameter 4-183
FORMSNO and CARDNO Parameter 4-183
Parameter Descriptions—AUTHORIZE 4-183
LIST Parameter 4-183
Parameter Descriptions—CLASSIFY 4-184
PROGRAM Parameter 4-184
UNIT Parameter 4-184
CLASS Parameter 4-184
LIBRARY Parameter 4-184
PACK Parameter 4-184
Spool File Considerations and Restrictions 4-184
FILE Requirements 4-184
Partition Size Requirements 4-184
Copy the Entire Spool File (COPYSP) 4-184
Copy Selected Job Steps from the Print Queue (COPYPRTQ) 4-184.1
Copy Selected Job Steps from the Punch Queue (COPYPCHQ) 4-190
Copy Jobs To or From the Reader Queue (COPYRDRQ) 4-192
Read Control Statements from a File (COPYCTRL) 4-194
Copy a Display of the Status of the Spool Queues (DISPLAY) 4-194
Restore Print or Punch Queue Records From a File (RESTORE) 4-195
Copy Selected Job Steps From One Spool File to Another (COPYQ) 4-195
CHANGE THE AUTHORIZ FILE (AUTHORIZE) . . . 4-196
Authorization Fields 4-196
Create Authorization Record 4-196.2
Change Authorization Record 4-196.2
Delete Authorization Record 4-196.2
ASSIGN A CLASS NUMBER TO A PROGRAM (CLASSIFY) 4-196.3
OCL Considerations 4-196.5
Examples 4-196.5
Using the Spool File Copy Program Under CCP . . . 4-202.1
Program Request 4-202.2
User Authorization 4-202.2
Using the Spool File Copy Program from a Terminal . 4-202.3
Responding to Error Messages 4-203
Placing Jobs on the Reader Queue from a Terminal 4-203
Displaying the Spool Queues 4-203
CCP Assignment Set 4-204
How to Request $QCOPY From a Terminal . . . 4-205
Examples 4-206
Considerations for Terminating the Spool File Copy Program Under CCP 4-206
RECOVER INDEX PROGRAM—$RINDX 4-208
Program Description 4-208
File Identification 4-209
OCL Considerations 4-210
Considerations and Restrictions 4-210
Examples 4-211
REASSIGN ALTERNATE TRACK PROGRAM—$RSALT 4-212
Program Description 4-212
Control Statement Summary 4-212
Parameter Summary 4-213
Parameter Descriptions 4-213
UNIT Parameter 4-213
PACK Parameter 4-213
SIMULATION AREA PROGRAM—$SCOPY . . . 4-216
Program Description 4-216
Control Statement Summary 4-217
Parameter Summary 4-219
Parameter Descriptions—COPYAREA 4-222
FROM and TO Parameters 4-222
PACK Parameter 4-222
AREA Parameter 4-222
TONAME Parameter 4-222
SYSTEM Parameter 4-222
Parameter Descriptions—CLEAR 4-222
FROM Parameter 4-222
PACK Parameter 4-222
AREA Parameter 4-222
CLRNAME Parameter 4-222
ID Parameter 4-222
TYPE Parameter 4-223
Parameter Descriptions—NEWNAME 4-223
TO Parameter 4-223
PACK Parameter 4-223

AREA Parameter 4-223
ID Parameter 4-223
SYSTEM Parameter 4-223
CLRNAME Parameter 4-223
Parameter Descriptions—COPYIPL 4-223
FROM and TO Parameters 4-223
PACK Parameter 4-223
TONAME Parameter 4-223
Parameter Descriptions—NAMES 4-223
PRINT Parameter 4-224
Parameter Descriptions—MOVE 4-224
FROM and TO Parameters 4-224
PACK Parameter 4-224
AREA Parameter 4-224
TONAME Parameter 4-224
OCL Considerations 4-224
Examples 4-225
TAPE INITIALIZATION PROGRAM—$TINIT 4-225
Program Description 4-233
Control Statement Summary 4-233
Parameter Summary 4-234
OCL Considerations 4-235
Printout of Volume Label 4-236
Messages for Tape Initialization 4-236
Meaning of Volume Label Information 4-236
TAPE ERROR SUMMARY PROGRAM—$TVES 4-237
Program Description 4-239
Error Logging Format 4-239
OCL Considerations 4-240
VTOC SERVICE PROGRAM—$WVTOC 4-240
Program Description 4-241
Control Statement Summary 4-241
Parameter Summary 4-241
Parameter Descriptions 4-241
PACK Parameter 4-241
UNIT Parameter 4-241
OCL Considerations 4-241
Examples 4-242
4-242

APPENDIX A. IBM SYSTEM/3 STANDARD CHARACTER SET A-1

APPENDIX B. CALCULATING FILE SIZE B-1
Data Area Track Requirements B-1
Index Area Track Requirements B-4
Track Usage for Index Files B-4
Core Index B-6
Calculating File Sizes (Main Data Area)—Summary B-7
Determining the Number of Tracks in a Sequential or Direct File B-7
Determining the Number of Tracks in an Indexed File (Main Data Area) B-7
Determining the Number of Tracks of Disk Track Index B-7
Converting Cylinder/Track to Track Number B-7
Converting Track Number of Cylinder/Track B-8

APPENDIX C. OPERATOR CONTROL COMMANDS (OCC) C-1
OCC Summary C-1
Information About Syntax Illustrations C-1

APPENDIX D. SUBR15—LIBRARY ENTRY RETRIEVAL SUBROUTINE D-1
Linking SUBR15 with RPG II D-2
Linking SUBR15 with Assembler D-5
Error Identification D-6

APPENDIX E. TRANSACTION LOGGING—$TRLOG E-1
Using Transaction Logging E-1
Tape Considerations E-1
Loading $TRLOG E-1
Controlling $TRLOG E-2
Operating Considerations E-2
Programming Considerations E-2

APPENDIX F. PROGRAM REFERENCE INFORMATION F-1

How to Use This Manual

This publication contains four parts. Part 1 describes Operation Control Language (OCL) statements, Part 2 describes the system concepts and facilities, Part 3 describes the format of a 3340 and 3344 volume, and Part 4 describes the system service programs.

PART 1

Refer to Part 1 if you want to know:

- What an OCL statement is
- What each OCL statement is used for (function)
- Where each OCL statement is placed in relation to others and when it is needed (placement)
- How each OCL statment must be coded (format)
- What each OCL statement must contain (contents)

PART 2

Refer to Part 2 if you want to know about:

- Model 15 programming concepts
- Files and file services
- Library facilities and concepts
- System operation overview
- Multiprogramming and spooling overview
- System control program facilities

PART 3

Refer to Part 3 if you want to know about:

- The format and storage capacity of a 3340 or 3344 volume
- Simulation areas
- Main data areas
- Alternate tracks
- Cylinder 0 format

PART 4

Refer to Part 4 if you want to know about:

- Which system service programs are supplied with the system
- The function of each system service program
- The operational control language (OCL) statements and control statements applicable to each system service program

APPENDIXES

Refer to the appendixes if you want information about:

- Standard System/3 character set
- Calculation of file sizes
- Operator control commands (OCC)
- How to retrieve library entries with SUBR15
- How to log transactions
- Individual programs associated with SCP (system control programming 5704-SC2)

This page is intentionally left blank.

System/3 Model 15D features a processing unit that allows the attachment of a 3340 Direct Access Storage Facility and a 3344 Direct Access Storage. With a 3340A2 and 3344B2, the maximum online disk storage is approximately 506 megabytes.

When compared to a System/3 Model 15A, Model 15B, or Model 15C, the Model 15D has a faster instruction cycle time for certain non-I/O instructions. The purpose of the faster cycle time is to complement the requirements of the enhanced programming support.

System/3 Model 15D is supported by a multiprogramming system control program (Program Number 5704-SC2) that resides on a simulation area of a direct access storage device. It provides functions that are not available on other System/3 models.

Three program partitions are supported. The scheduling and controlling of programs in the partitions is controlled by operation control language (OCL) statements and operator control commands (OCC). See Appendix C for a summary of the commands.

Greater online library capacity is available because each partition can directly access three unique simulation areas and shares a common simulation area (the IPL area). Simulation areas are assigned by the user; reassignments for simulation areas other than the IPL area require an appropriate OCL statement. (More information about simulation areas is included in Part 3 of this manual.)

Input job streams are made up of jobs and job steps. A job is one or more LOAD/RUN or CALL/RUN sequences grouped together to execute in sequence and perform a specific function. A job step is one LOAD/RUN or CALL/RUN sequence. A JOB OCL statement must be used to group job steps together to form a job. See Part 2 for a discussion of jobs and job steps.

As the system processes an input stream, jobs are processed in job mode. Job steps not contained within a specified job are processed, by the system, in step mode. See Part 2 for a discussion of job and step mode.

Program support provides a spooling function for certain input and output operations. Spooling places jobs from the input stream in a special area on disk called a reader queue. Jobs are transferred from disk by spooling to the partitions as required for execution.

Printed and/or punched output is placed in queues on disk during execution of job steps. Printing or punching to the associated devices is performed by spooling. Spooling provides greater flexibility during job scheduling and removes many I/O device conflicts between partitions. See Part 2 for more information on spooling.

Support for directly attached 3741 Data Station/Programmable Work Station is similar to that for a card reader or card punch. In this manual, unless otherwise noted, references to card I/O also apply to the directly attached 3741.

Part 1. OCL Statements

WHAT IS OCL?

Operation Control Language (OCL) is one means of communicating with the system. Operator control commands (OCC) are another means of communicating with the system. See Appendix C for a summary of OCC. You must provide a set of OCL statements for each program you want to run. Based on the information supplied in these statements, the system loads and executes your programs or performs system service functions.

You can supply OCL statements in four ways: (1) by punching the statements into cards, which are then read by the system; (2) by using the CRT/Keyboard to key the statements directly into the system; (3) by keying the statements onto a diskette, which is then read by the system; (4) by using procedures.

After the system reads a set of OCL statements for a program, it runs the program. When the program ends, the system reads the next set of statements and runs that program. This cycle is repeated until all OCL statements have been read and the corresponding programs have been run.

The running of your program is controlled by system control programs. System control programs must be in main storage before your jobs can be run. These programs must be located in simulation areas.

A procedure called initial program load (IPL) initiates the loading of selected system control programs. IPL must be performed by the operator after the system power-on sequence. Other system control programs are brought into main storage, as required, during program loading and execution.

OCL and the Job Stream

The OCL statements you supply form the basis of the job stream. If your program requires data from the system input device (the device used to read OCL statements), the data must follow the corresponding OCL. The job stream can contain programs and program data as well as OCL statements. (Figure 1-1 shows an input job stream.)

You can also store sets of OCL statements for your programs outside the job stream in a source library on disk. These sets are called procedures. You can instruct the system to merge procedures into the job stream, which eliminates recoding frequently-used statements. (See *Procedures* in Part 2 of this manual.)

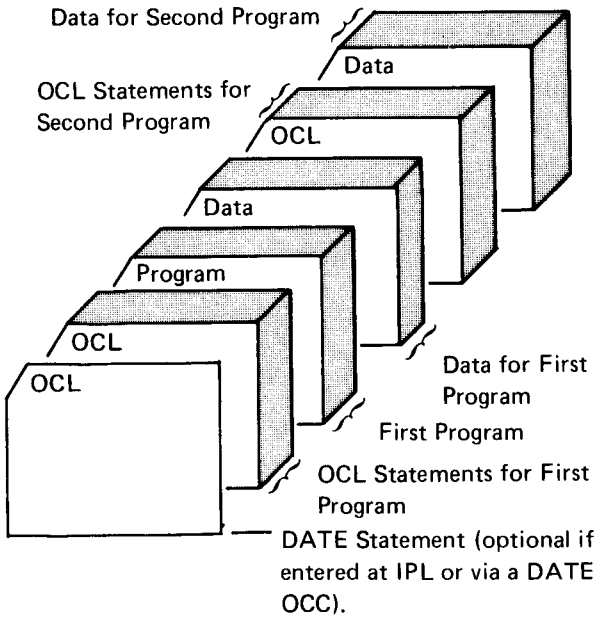


Figure 1-1. Input Job Stream

ORGANIZATION OF PART 1

Part 1 is divided into:

- Coding Rules – Defines the general contents of the OCL statements and explains the rules for writing the statements.
- Statement Descriptions – Explains the functions, format, and contents of each OCL statement, and where each statement may be used in the job stream.
- Statement Examples – Presents and explains a job stream containing most of the OCL statements.

CODING RULES

Types of Information

OCL statements contain, at most, two types of information: a statement identifier and parameters. The statement identifier distinguishes one statement from another; the parameters supply additional information. The following example shows the format of an OCL statement.

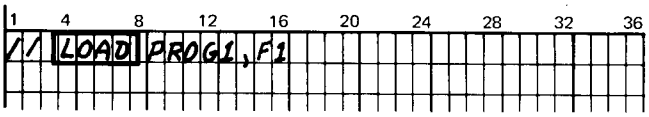
Identifier	Parameter 1, Parameter 2, ..., Parameter n

Statement Identifiers

Every OCL statement needs one of these statement identifiers:

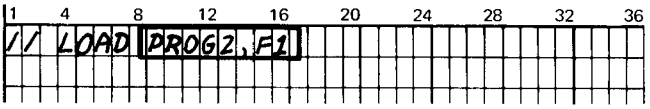
ASSIGN	IMAGE	PUNCH
BSCA	JOB	READER
CALL	LOAD	RUN
COMPILE	LOG	SWITCH
DATE	NOHALT	/&
FILE	PAUSE	/.
HALT	PRINTER	* (asterisk)

The word LOAD is an example of a statement identifier.

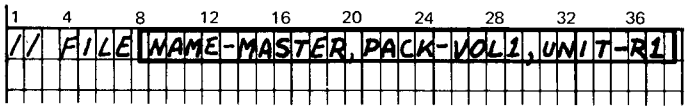


Parameters

Some statements need parameters; others do not (see *Statement Descriptions* for an explanation). Parameters can be either codes or data. A code is a word or group of characters that has a certain meaning. Data is information such as the names, locations, and lengths of files on disk. (See *Statement Descriptions* for data and code restrictions on parameters.) In the following example, PROG2 is the name of an RPG II object program, and F1 is a 5444 unit code that is assigned to a simulation area. PROG2 is a data parameter and F1 is a code parameter.



Some statements require certain words in parameters to tell one parameter from another. The words are called keywords. Parameters containing keywords are called keyword parameters. (In the following example, NAME-MASTER, PACK-VOL1, and UNIT-R1 are keyword parameters. NAME, PACK, and UNIT are keywords. MASTER and VOL1 are data parameters. R1 is a code parameter.) A hyphen is always required between the keyword and the code or data parameter.



General Coding Rules

In Part 1 of this manual, the numbers that appear above statement formats and examples indicate the card columns or line positions occupied by the statements. In statement formats, special characters (such as //) and words written in capital letters represent information that must be used exactly as shown. Words written in small letters (such as code, program-name, and unit) represent information that you must supply.

Special Meaning of Capital Letters, Numbers, and Special Characters

Capitalized words and letters, numbers, and special characters have special meanings in OCL and statement descriptions.

Words or letters that are not capitalized indicate that you must supply a value that applies to the job you are doing. The values that can be used are listed in the parameter summaries.

Braces { } and brackets [ ] sometimes appear in parameters shown in statement summaries and parameter summaries. They are not part of the parameter; they simply indicate a choice of values to complete the parameter. You *must* choose one of the values surrounded by braces; you *may* choose a parameter surrounded by brackets or omit that parameter entirely. Underscoring of one value enclosed by braces indicates the default. If you specify the keyword of a parameter, you must complete the parameter by supplying the code or data even though a default is indicated.

For example:

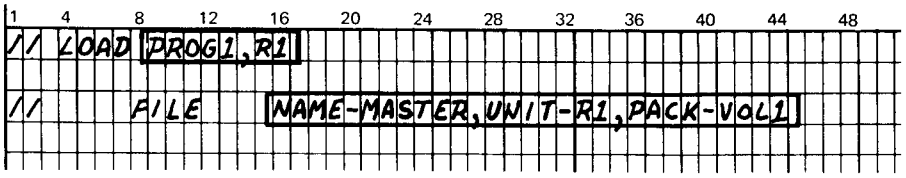
- [RECL-{80 / 96}] means that if you do not specify this parameter, the system will select RECL-96. If you specify the keyword RECL, you must also supply one of the values (80 or 96).
- RETAIN-{T / P} means that you must specify either RETAIN-T or RETAIN-P.
- [,BLKL-block length] means that the block length parameter may be omitted entirely.

Statements Beginning with //

The rules for coding the statements are as follows (the term position refers to either record column or line position):

- The // must be placed in positions 1 and 2. The *, /&, and /. statements are exceptions and must start in position 1. (See *Statement Descriptions* for *, /&, and /. statements.)
- There must be one or more blanks between the // and the word that forms the statement identifier (DATE, RUN, CALL, etc.). Exceptions are the JOB statement, which must have a jobname immediately following the //, and the LOAD statement, which may have a stepname immediately following the //.
- There must be one or more blanks between the end of the statement identifier and the first parameter.
- If you need more than one parameter, use a comma to separate them. No blanks are allowed within or between parameters. For the exception to this rule, see the description for the HIKEY parameter under *FILE Statement (Multivolume Disk Files).* Anything following the first blank after the last parameter is considered a comment (see *Comments*).
- If you are writing keyword parameters, place the keyword first, and use a hyphen to separate the keyword from the code or data parameter.
- If the parameter is not a keyword parameter, write the parameters in the order in which they are discussed in this manual.

(In the following example, the statement identifiers are LOAD and FILE. The parameters are PROG1, R1, NAME-MASTER, UNIT-R1, and PACK-VOL1.)



Statement Length

OCL statement length is as follows:

Device	Number of Characters
MFCU	96
MFCM	80
1442	80
2501	80
CRT/Keyboard	96
3741	96

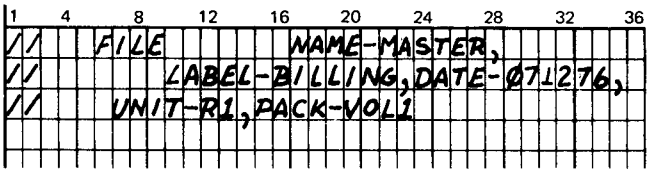
Continuation

The only OCL statements that may exceed 80 or 96 characters, including blanks and comments, are FILE, COMPILE, PUNCH, and PRINTER. Otherwise, each record you use must not exceed 80 or 96 characters. (Data for the IMAGE statement requires continuation for the cards or lines containing the chain image characters, but the data follows different continuation rules. See *IMAGE Statement* under *Statement Descriptions* for more information.)

The continuation rules are as follows:

- There must be a comma after the last parameter in every record except the last parameter in the OCL statement. The comma, followed by a blank, tells the system that the statement is continued in the next record.
- Each new record must begin with a // in positions 1 and 2.
- There must be one or more blanks between the // and the first parameter in the record. (See *HIKEY Parameter* under *FILE Statement [Multivolume Disk Files]* for the exception to this rule.)

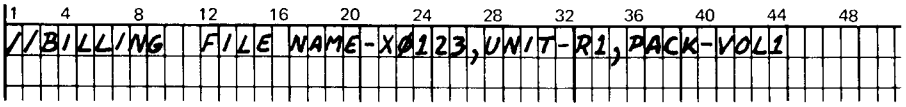
The following illustration is an example of the continuation rules:



Comments

Your statements can include comments in the following places:

- Following the // in any statement beginning with //. The comment must begin in position 3 immediately following the //. You can use up to 8 characters without blanks. There must be one or more blanks between the comment and the word forming the statement identifier. (In the following example, the word BILLING is the comment.) Comments of this type, when used in a JOB or LOAD statement, are treated as jobname and stepname, respectively.

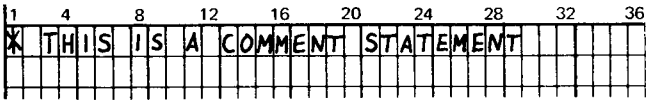


- After the last parameter. There must be one or more blanks between the last parameter and your comment. The comment can be any combination of characters except dashes. If the statement is continued in subsequent records, you can place comments after the last parameter in any of the records.

- After statements without parameters. There must be one or more blanks between the statement identifier and your comment. Examples of statements without parameters are: /&, // PAUSE, and // RUN.

In addition to writing comments within your OCL statements, you can include entire records of comments. The OCL comment statement is provided for that purpose (there must be an * in position 1 followed by the comment). For more information about the comment statement, refer to **(Comments) Statements* under *Statement Descriptions.*

The following is an example of a comment statement:



Spooling and Multiprogramming Considerations

All OCL statements can be used in a spooled input job stream, and all can be used in any partition.

Note: The DATE, IMAGE, LOG, PRINTER, PUNCH, READER, and /. OCL statements require special consideration when used in a spooled input job stream.

Statement Descriptions

The following information is given separately for each OCL statement in this section:

- Function of the statement
- Placement of the statement in regard to other statements, and the circumstances under which the statement is needed
- Format of the statement
- Contents of the statement (explaining the parameters that can be used in the statement)
- Spooling considerations for each statement

Figure 1-2 gives the function, placement, and restrictions on use for each OCL statement. Figure 1-3 describes the contents of the OCL statements and is meant for reference only. When using Figure 1-3, remember that words written in small letters, such as *filename* or *value,* require a choice on your part, depending on the functions you want the statement to perform. Capitalized parameters must be coded along with the data or code parameter. (Figure 1-3 shows which parameters are available.)

If you are not familiar with an entry, or you do not know when to use or omit it, refer to the proper statement in the remainder of this section.

Statement	Function	Placement		Coding Notes
		Statement Appears In Job Stream	Statement Appears In a Procedure	
// ASSIGN	Allows reassignment of a 5444 unit code (R1, F1, R2, F2) to a simulation area.	Anywhere among the OCL statements	Must precede the RUN statement (if RUN is used).	System pack (R1 or F1) cannot be reassigned.
// BSCA	Changes the BSCA line number.	Must follow LOAD or CALL statement and precede the RUN statement.	Must follow the LOAD statement and precede the RUN statement (if RUN is used).	None
// CALL	Identifies procedure to be merged into job stream and disk containing the source library from which to read the procedure.	Must precede the RUN statement. Must follow the JOB statement when the system is operating in job mode.	Must precede the RUN statement (if RUN is used).	1. If found in a procedure, indicates nested procedures. No more than nine levels of nested procedures allowed. 2. Must not be between LOAD and RUN or CALL and RUN.
// COMPILE	Supplies information about the program to be compiled or assembled to the compiler and the linkage editor.	Must follow LOAD or CALL statement and precede the RUN statement.	Must follow the LOAD statement and precede the RUN statement (if RUN is used).	Only one COMPILE statement allowed per job step.
// DATE	Changes system date and partition dates.	Must precede the first JOB, CALL, or LOAD statement.	Not applicable	None
	Changes partition date for job.	Must follow the JOB statement and precede a LOAD statement (either before the first step or between steps).	Before LOAD statement.	None
	Changes partition date for step.	Must follow the LOAD or CALL statement and precede the RUN.	After LOAD and before RUN.	Cannot be entered if DATE was used to change partition date for job.
// FILE	Supplies information about a file to the system.	Must follow LOAD or CALL statement and precede the RUN statement.	Must follow the LOAD statement and precede the RUN statement (if RUN is used).	Required for every new file created and for existing files being used.

Figure 1-2 (Part 1 of 5). Table of OCL Statements

Statement	Function	Placement		Coding Notes
		Statement Appears In Job Stream	Statement Appears In a Procedure	
// HALT	Instructs system to halt when program ends; cancels the effect of nohalt mode.	Anywhere among the OCL statements.	Must precede the RUN statement (if RUN is used).	None
// IMAGE	Tells the system to replace the chain-image area with characters indicated in the data records that are read from the system input device or read from the source library.	Anywhere among the OCL statements.	Must precede the RUN statement (if RUN is used).	Required if the printer chain has been changed.
// INCLUDE	Identifies the entry in the source library that contains the OCL statements to be merged into the job stream.	Anywhere among the OCL statements.	Must precede the RUN statement (if RUN is used).	1. If system service program control statements follow the RUN statement in the source member, they are placed in the SWA and read from there by the system service program. 2. Must not be between CALL and RUN.
//jobname JOB	Allows you to run related job steps together to ensure they are run sequentially.	Must precede the first LOAD or CALL statement for a job.	Cannot be used in a procedure.	Places a partition in job mode. Required whenever spooling is active.

// LOAD
or
//stepname LOAD
	Identifies the program to be run and indicates the disk that contains the object library from which it is to be loaded.	Must precede the RUN statement. Must follow the JOB statement when the system is operating in job mode.	Must precede the RUN statement (if RUN is used).	None

Figure 1-2 (Part 2 of 5). Table of OCL Statements

Statement	Function	Placement		Coding Notes
		Statement Appears In Job Stream	Statement Appears In a Procedure	
// LOAD * or //stepname LOAD *	Indicates that after the RUN statement is processed, the object program will be loaded from the system input device or from the file indicated on the specified unit.	Must precede the RUN statement. Must follow the JOB statement when the system is operating in job mode.	Must precede the RUN statement (if RUN is used). Object program will be read from the system input device.	A LOAD * program cannot be loaded if another LOAD * program (with overlays) is executing in another partition.
// LOG	Changes the device used for displaying system messages and controls page ejection before EJ and ES and after EJ.	Anywhere among the OCL statements.	Must precede the RUN statements (if RUN is used).	Applies only to the partition in which it was entered.
// NOHALT	Instructs system to continue without stopping when a program ends and/or sets the severity level of halts.	Anywhere among the OCL statements.	Must precede the RUN statement (if RUN is used).	None
// PAUSE	Causes OCL processing to stop in order to give the operator time to perform a function. Operator must restart OCL processing.	Anywhere among the OCL statements.	Must precede the RUN statement (if RUN is used).	This is the only OCL statement displayed on the CRT.
// PRINTER	Enables you to describe the functions performed by the system print device and control options related to print spooling.	Anywhere among the OCL statements.	Must precede the RUN statement (if RUN is used).	None

Figure 1-2 (Part 3 of 5). Table of OCL Statements

Statement	Function	Placement		Coding Notes
		Statement Appears In Job Stream	Statement Appears In a Procedure	
// PUNCH	Enables you to describe the functions performed by the system punch device and control options related to punch spooling.	Anywhere among the OCL statements.	Must precede the RUN statement (if RUN is used).	None
// READER	Changes the system input device used to read OCL statements.	Must precede LOAD or CALL statement.	Must precede the LOAD statement (if LOAD is used).	If used in a procedure, the system input device is changed when the READER statement is processed; but OCL statements are not read from the new system input device until the procedure is completely executed.
// RUN	Indicates the end of the OCL statements for a job step and tells system to run the program.	Must follow the LOAD or CALL statement and be the last OCL statement for a job step.	If used, must follow the LOAD statement and be the last OCL statement in the procedure.	Required in the job stream for each job step which is to be run.
// SWITCH	Used to set one or more external indicators on or off or to leave the indicator as it is.	Anywhere among the OCL statements.	Must precede the RUN statement (if RUN is used).	Only one switch statement allowed between LOAD or CALL and RUN.
/&	Acts as a delimiter between job steps.	Recommended as the last OCL statement of a job step.	Not allowed in a procedure.	Not allowed in a procedure.

Figure 1-2 (Part 4 of 5). Table of OCL Statements

Statement	Function	Placement		Coding Notes
		Statement Appears In Job Stream	Statement Appears In a Procedure	
/.	1. With spooling active, acts as a delimiter between jobs. Causes end of job.	Recommended as the last OCL statement of a job.	Not allowed in a procedure.	Not allowed in a procedure.
	2. With input spooling active, two consecutive /. statements indicate end of spooled input.	Last two OCL statements in the input job stream.	Not allowed in a procedure.	Not allowed in a procedure.
	3. With spooling inactive, indicates end of job mode. Causes end of job. Next OCL statement may start job step mode.	Last OCL statement	Not allowed in a procedure.	Not allowed in a procedure.
* (Comment)	Used to explain the job, to write a time stamp to the SHA, or to give the operator instructions; does not affect program operation.	Anywhere among the OCL statements.	Anywhere among the OCL statements.	Comments are not displayed on the CRT.

Figure 1-2 (Part 5 of 5). Table of OCL Statements

Statement	Parameter	Code	Meaning of Code
// ASSIGN	R1- R2 F1 F2	D1A, D1B, D1C, D1D, D2A, D2B, D2C, D2D, D3A, D3B, D3C, D3D, D3E, D3F, D3G, D3H, D4A, D4B, D4C, D4D, D4E, D4F, D4G, D4H	Assign (by partition) a 5444 unit code to a simulation area code. (Refer to *ASSIGN Statement*.)
	AREA	AREA-name	Name of simulation area to be assigned.
	PACK	PACK-name	Name of main data area associated with simulation area.
// BSCA	LINE	LINE-1 2	Change all BSCA DTF line codes to the line number specified.
// CALL	procedure name	name	Name that identifies the procedure in the source library.
	unit		Specifies the simulation area that contains the procedure.
		5444 unit code	Unit code for simulation area. Possible codes are R1, F1, R2, F2.
	switch characters	xxxxxxxx	Specifies 8 switch characters that are compared with the eight external indicators. Possible characters are 0, 1, or X.
// COMPILE	SOURCE UNIT	SOURCE-name UNIT-5444 unit code	Name of source program. Specifies the simulation area that contains the source library. Possible codes are R1, F1, R2, F2.
	OBJECT	OBJECT-5444 unit code	Specifies the simulation area that is to receive the object program. Possible codes are R1, F1, R2, F2.
	LINKADD	LINKADD–4000 8000	Linkage Editor: start address (hexadecimal).
	ATTR	ATTR–MRO	Requests that the object program be link-edited to use REMAP mode, memory resident overlays.
		MOV	Requests that the object program be link-edited to use MOVE mode, memory resident overlays.
// DATE	date	mmddyy ddmmyy	System date or partition date (domestic date format). System date or partition date (World Trade date format).

Figure 1-3 (Part 1 of 7). Table of Parameters

Statement	Parameter	Code	Meaning of Code
// FILE (disk files)	NAME	NAME-filename	Name the program uses to refer to the file.
	UNIT	UNIT-5444 unit code	Specifies the simulation area that contains or will contain the file. Possible codes are R1, F1, R2, F2.
		UNIT-main data area code	Specifies the main data area that contains or will contain the file. Possible codes are D1, D2, D3 or D31, D32, D33, D34, D4 or D41, D42, D43, D44.
	PACK	PACK-name	Name of area that contains or will contain the file.
	LABEL	LABEL-filename	Name by which your file is identified or will be identified on disk.
	RECORDS or	RECORDS-number or	Approximate number of records for the file.
	TRACKS	TRACKS-number	Number of tracks required by the file.
	LOCATION	LOCATION-track number	Track number on which file begins or is to begin (simulation area only).
		LOCATION-cylinder number	Cylinder number on which file begins or is to begin. Track assumed zero (main data area only).
		LOCATION-cylinder number/track number	Cylinder number, track number on which file begins or is to begin (main data area only).
	RETAIN	RETAIN-T	Temporary file.
		S	Scratch file.
		P	Permanent file.
	DATE	DATE-mmddyy ddmmyy	Tells the system the date the file was created.
	HIKEY	HIKEY-'highest unpacked key fields allowed'	List of highest unpacked key fields allowed on each pack of an indexed multivolume file (main data area only).
		—or—	
		HIKEY-P 'highest packed keyed fields allowed'	List of highest packed key fields allowed on each pack of an indexed multivolume file (main data area only).
	VERIFY	VERIFY-YES	Verify disk write operations for this file (main data area only).
		NO	Do not verify disk write operations for this file.
	SHARE	SHARE-YES	Allow file sharing between partitions if access methods are compatible.
		NO	Do not allow file sharing. Multiple references to the same file within one program are not allowed if SHARE-NO is specified. If parameter is omitted, SHARE-YES is assumed.

Figure 1-3 (Part 2 of 7). Table of Parameters

Statement	Parameter	Code	Meaning of Code
// FILE (tape file)	NAME	NAME-filename	Name the program uses to refer to the file.
	UNIT	UNIT-T1	Where the tape that contains or will contain the file is mounted.
		T2	
		T3	
		T4	
	REEL	REEL-nnnnnn	Name of the labeled tape that contains or will contain the file.
		NL	The tape is not labeled.
		NS	The tape contains non-standard labels (input only).
		BLP	A standard labeled tape is mounted. Bypass label processing (input only).
	LABEL	LABEL-name or LABEL-'character string'	Name by which your file is identified on tape.
	DATE	DATE-mmddyy	The date the file was created.
		ddmmyy	
	RETAIN	RETAIN-nnn	The number of days a file should be retained before it expires.
	BLKL	BLKL-block length	The number of bytes in a physical block on tape.
	RECL	RECL-record length	The number of bytes in a logical record.
	RECFM	RECFM-F	Fixed length, unblocked records.
		V	Variable length, unblocked records.
		D	Variable length, unblocked D-type ASCII records.
		FB	Fixed length, blocked records.
		VB	Variable length, blocked records.
		DB	Variable length, blocked, D-type ASCII records.
	END	END-LEAVE	The tape remains in position after the file is processed.
		UNLOAD	The tape is rewound and unloaded after processing.
		REWIND	The tape is rewound after processing.
	DENSITY	DENSITY-200	The tape will be written at 200 bpi (bits per inch) density.
		556	The tape will be written at 556 bpi density.
		800	The tape will be written at 800 bpi density.
		1600	The tape will be written at 1,600 bpi density.
			Default for 7-track is 800 bpi.
			Default for 9-track is 1,600 bpi.
	ASCII	ASCII-YES	An ASCII file is being processed or created.
		NO	An EBCDIC file is being processed or created.
	DEFER	DEFER-YES	The tape volume will be mounted later.
		NO	The tape is presently mounted.
	CONVERT	CONVERT-ON	Data read from or written to a 7-track tape file will be converted.
		OFF	Data read from or written to a 7-track tape file will not be converted.
	TRANSLATE	TRANSLATE-ON	Data read from or written to a 7-track tape file will be translated.
		OFF	Data read from or written to a 7-track tape file will not be translated.

Figure 1-3 (Part 3 of 7). Table of Parameters

Statement	Parameter	Code	Meaning of Code
// FILE (tape) (continued)	PARITY	PARITY-EVEN	The 7-track tape file will be read or written in even parity.
		ODD	The 7-track tape file will be read or written in odd parity.
	SEQNUM	SEQNUM-nnnn	File sequence number can be 0001 to 9999.
		X	Prepositioned file.
// FILE (device independent card, diskette, or printer files)	NAME	NAME-filename	Name the program uses to refer to the file.
	UNIT	UNIT-MFCU1	Primary hopper of 5424 MFCU.
		MFCU2	Secondary hopper of 5424 MFCU.
		MFCM1	Primary hopper of 2560 MFCM.
		MFCM2	Secondary hopper of 2560 MFCM.
		1442	1442 Card Read Punch.
		2501	2501 Card Reader.
		3741	3741 Data Station/Programmable Work Station.
		1403	1403 Printer.
		3284	3284 Printer.
		READER	Use the partition's assigned system input device.
		PRINTER	Use the partition's assigned system print device.
		PUNCH	Use the partition's assigned system punch device.
	PRINT	PRINT-YES	Interpreting is to be done on punch files.
		NO	Interpreting is not to be done on punch files.
	RECL	RECL-record length	Number of bytes in a logical record (3741 only).
// HALT	None		
// IMAGE	format	HEX	Characters from system input device are in hexadecimal form.
		CHAR	Characters from system input device are in EBCDIC form.
		MEM	Characters are from the source library.
	number	value	Number of new characters.
	name	name	Identifies the source member containing the characters in the source library.
	unit	5444 unit code	Specifies the simulation area that contains the source library. Possible codes are R1, F1, R2, F2.
// INCLUDE	procedure name	name	The name of the procedure that contains the OCL to be merged.
	unit	5444 unit code	Specifies the simulation area that contains the procedure. Possible codes are R1, F1, R2, F2.
	switch characters	xxxxxxxx	Specifies 8 switch characters that are compared with the eight external indicators. Possible characters are 0, 1, X.

Figure 1-3 (Part 4 of 7). Table of Parameters

Statement	Parameter	Code	Meaning of Code
//jobname JOB	PRIORITY	PRIORITY-0 1 2 3 4 5	Specifies a job's priority on the reader queue and on the output queues unless overridden by a PRIORITY parameter on a PRINTER or PUNCH statement.
	CORE	CORE-size	Specifies amount of main storage required to execute the largest step of a job.
	SPOOL	SPOOL-YES NO	Indicates whether spooling can or cannot be used for the job. Default is YES.
	PARTITION	PARTITION-1 2 3 A B C D	Specifies the partition in which a spooled job should be executed. A means 1 or 2; B means 1 or 3; C means 2 or 3; D means 1, 2, or 3.
	QCOPY	QCOPY-YES NO	Allows (QCOPY-YES) or disallows (QCOPY-NO) the spool file copy program ($QCOPY) to access this job on the reader queue.

// LOAD
or
//stepname
LOAD
	asterisk
	program-name
	unit
	switch characters
		*
		name
		main data area code
		xxxxxxxx
			Program is to be loaded from the system input device.
			Identifies the file that contains the object program.
			Specifies the main data area that contains the file.
			Specifies 8 switch characters that are compared with the eight external indicators. Possible characters are 0, 1, X.

// LOAD
or
//stepname
LOAD
	program-name
	unit
	switch characters
		name
		5444 unit code
		xxxxxxxx
			Name of program that is to be loaded from an object library on disk.
			Specifies the simulation area that contains the program. Possible codes are R1, F1, R2, F2.
			Specifies 8 switch characters that are compared with the eight external indicators. Possible characters are 0, 1, X.

// LOG	device	CONSOLE 1403 3284	Log to CRT and system history area on system pack. Log to CRT, the 1403 printer, and the system history area on the system pack. Log to CRT, the 3284 printer, and the system history area on the system pack.
	mode	EJECT[1] NOEJECT[1]	Eject a page before ES and EJ and after EJ. Do not eject a page before ES and EJ and after EJ.

[1] When you use the spool print writer, an eject occurs at the start of every job step, regardless of the mode specified in the LOG statement.

Figure 1-3 (Part 5 of 7). Table of Parameters

Statement	Parameter	Code	Meaning of Code
// NOHALT	SEVERITY	SEVERITY-1 2 4 8	Tells the system to select default options for error halts.
// PAUSE	None		
// PRINTER	DEVICE	DEVICE-1403 3284	1403 Printer is used as the system print device. 3284 Printer is used as the system print device.
	LINES	LINES-number	Specifies the number of print lines per page.
	FORMSNO	FORMSNO-forms type	Informs the operator which forms type should be mounted on the printer, and determines the forms type of the spooled printed output.
	COPIES	COPIES-number	With spooling active, allows you to obtain more than one copy of each job step's printed output.
	DEFER	DEFER-YES NO	Allows you to begin printing a job step's spooled output before the job step completes execution (DEFER-NO). Default is DEFER-YES.
	CLOSE	CLOSE-YES NO	Allows you to control when print spool will close an intercepted job step on the spool file.
	QCOPY	QCOPY-YES NO	Allows you to prevent spool file print records from being copied by the spool file copy program.
	ALIGN	ALIGN-YES NO	Allows you to perform forms alignment for spooled printed output (ALIGN-YES). Default is ALIGN-NO.
	PRIORITY	PRIORITY-0 1 2 3 4 5	Specifies a job step's priority on the spool print queue. Default is the priority of the job at the time it is executed.
// PUNCH	DEVICE	DEVICE-MFCM1 MFCM2 MFCU1 MFCU2 1442 3741	Primary hopper of 2560 MFCM. Secondary hopper of 2560 MFCM. Primary hopper of 5424 MFCU. Secondary hopper of 5424 MFCU. 1442 Card Read Punch. 3741 Data Station/Programmable Work Station.
	CARDNO	CARDNO-card type	Tells the operator which card type to use for punching.
	COPIES	COPIES-number	With spooling active, allows you to obtain more than one copy of each job step's punched output.
	DEFER	DEFER-YES NO	Allows you to begin punching a job step's spooled output before the job step completes execution (DEFER-NO). Default is DEFER-YES.
	QCOPY	QCOPY-YES NO	Allows you to prevent spool file punch records from being copied by the spool file copy program.
	PRIORITY	PRIORITY-0 1 2 3 4 5	Specifies a job step's priority on the spool punch queue. Default is the priority of the job at the time it is executed.

Figure 1-3 (Part 6 of 7). Table of Parameters

Statement	Parameter	Code	Meaning of Code
// READER	code	CONSOLE MFCU1 MFCU2 MFCM1 MFCM2 1442 2501 3741	CRT/keyboard. Primary hopper of 5424 MFCU. Secondary hopper of 5424 MFCU. Primary hopper of 2560 MFCM. Secondary hopper of 2560 MFCM. 1442 Card Read Punch. 2501 Card Reader. 3741 Data Station/Programmable Work Station.
// RUN	None		
// SWITCH	indicator-settings	0 1 X	Set external indicator off. Set external indicator on. Leave external indicator as it is.
/&	None		
/.	None		
* (Comment) * TIME	None		Write a time stamp to the SHA (system history area).

Figure 1-3 (Part 7 of 7). Table of Parameters

ASSIGN Statement

Function

The ASSIGN statement allows the reassignment of the 5444 unit codes (F1, R1, F2, R2) to any of the supported simulation areas. The reassignment(s) applies only to the partition in which the ASSIGN statement is processed. The system pack cannot be reassigned. Therefore, a maximum of nine user-assigned simulation areas is allowed (three per partition). All reassignments remain in effect until another ASSIGN statement is processed or until another initial program load (IPL) is performed.

Placement

The ASSIGN statement can appear anywhere among the OCL statements. In a procedure, the ASSIGN statement must precede the RUN statement.

Format

// ASSIGN 5444 unit code – simulation area code [,AREA-name] [,PACK-name]

Contents

Possible 5444 unit codes are R1, F1, R2, F2.

Possible simulation area codes are:

3340 drive 1	D1A, D1B, D1C, D1D
3340 drive 2	D2A, D2B, D2C, D2D
3340 drive 3	D3E, D3A
3340 drive 4	D4E, D4A
3344 drive 3	
volume 1	D3E, D3A
volume 2	D3F, D3B
volume 3	D3G, D3C
volume 4	D3H, D3D
3344 drive 4	
volume 1	D4E, D4A
volume 2	D4F, D4B
volume 3	D4G, D4C
volume 4	D4H, D4D

AREA-name: This optional parameter is used to verify the name of the simulation area. The system checks this name against the actual simulation area name to ensure that the proper simulation area is being assigned. For information about how a simulation area is given a name, refer to *Simulation Area Program ($SCOPY).*

PACK-name: This optional parameter is used to verify the name of the main data area associated with the simulation area specified on the ASSIGN statement. The system checks this name against the actual main data area name to ensure that the proper main data area is online and in a ready state. For information about how a main data area is given a name, refer to *Disk Initialization Program ($INIT).*

Examples

// ASSIGN F2-D4C,R1-D3B,R2-D1D

For the partition in which the ASSIGN statement is processed, the following simulation area assignments are made:

- Simulation area D4C on volume 3 of 3344 drive 4 will be referenced as F2.
- Simulation area D3B on volume 2 of 3344 drive 3 will be referenced as R1.

- Simulation area D1D on 3340 drive 1 will be referenced as R2.
- The simulation area assignment for F1 was made during IPL.

  // ASSIGN R1-D3E,PACK-D3D3D3,AREA-D3ED3E

- For the partition in which the ASSIGN statement is processed, simulation area D3E on drive 3 will be referenced as R1.
- The name of the simulation area is verified as being D3ED3E (AREA-D3ED3E).
- The name of the main data area associated with the simulation area is verified as being D3D3D3 (PACK-D3D3D3).

Considerations and Restrictions

The reassignment of the simulation areas is effective immediately after the ASSIGN statement is processed.

Assume that an IPL is performed with the PROGRAM LOAD SELECTOR switch setting at DISK 1 F1. The system will assign F1 to D1A for all partitions; the unit code assigned to the system pack must not be used in the ASSIGN statement. Two different 5444 unit codes cannot be assigned to the same simulation area within a partition.

An ASSIGN statement is not processed when it is included among the OCL statements of a job step that is flushed.

If the PACK and/or AREA parameter is specified, only one simulation area can be re-assigned on each ASSIGN statement.

The name of the area (main data and/or simulation) is checked only if the appropriate parameter (PACK and/or AREA) is specified.

BSCA Statement

Function

The BSCA statement allows you to change all BSCA line specifications in your program. Therefore, you can use BSCA line 1 or 2 without recompiling the program. If the BSCA statement is not entered, the line specifications in the program are not changed.

Placement

The BSCA statement must follow the LOAD or CALL statement and precede the RUN statement.

Format

// BSCA parameter

Contents

The parameter LINE-code is a keyword parameter. The codes are as follows:

Code	Meaning
1	Change all BSCA line specifications to BSCA line 1.
2	Change all BSCA line specifications to BSCA line 2.

Spooling Considerations

None

CALL Statement

Function

CALL statements are needed only when you want to call a procedure from the source library.

To understand the function of the CALL statement, you must understand the relationship between the job stream and procedures. The job stream contains the OCL statements that control the system. The system reads the job stream from the system input device. Procedures are sets of OCL statements in a source library on disk. They have no effect on the stream until they are merged into the job stream.

You can modify the procedure identified by a CALL statement, by providing other OCL statements (procedure override statements, see *Changing Procedure Parameters*) after the CALL statement. These statements temporarily modify the procedure. The last statement of the CALL sequence must be a RUN statement. The RUN statement is required whether or not you supply other OCL statements. (Procedures are further explained in Part 2.)

Placement

CALL statements can be used in the job stream or in a procedure. They are, in effect, replaced by the procedures they identify. The CALL statement must precede the RUN statement. On spooled systems, it must follow the JOB statement. It must not be between LOAD and RUN or CALL and RUN.

Format

// CALL procedure-name,unit,switch characters (optional)

Contents

Procedure-name: The procedure-name is the name that identifies the procedure in the source library. You supply the procedure-name in the library maintenance control statements when you use that program to place the procedure in the library. (See *Library Maintenance Program* in Part 4 of this manual for restrictions on procedure-name.)

Unit: The unit parameter is a required code. The code identifies the simulation area that contains the procedure. Possible codes are R1, F1, R2, F2.

Switch characters: The switch characters (0, 1, and X) are optional. When you include them, you must supply 8 characters because they are compared with the eight external indicators. The system does a comparison for each position if the switch character is a 0 or 1. An X cancels the compare operation for that position only. The first (leftmost) switch character is compared with external indicator 1; then the second switch character is compared with external indicator 2; this process continues until the 8 switch characters and the eight external indicator positions are either compared or bypassed. If an equal condition exists, the procedure is called. Otherwise, an informational message is displayed and the job stream is flushed to the next step.

Example

The *Procedures* section in Part 2 contains CALL statement examples.

Spooling Considerations

None

COMPILE Statement

Function

The COMPILE statement tells the system where the source program to be compiled is located (if it is coming from a source library), and where the object program is to be placed. (An object program is the result of compiling or assembling a source program.) The COMPILE statement also specifies the options to the linkage editor.

Placement

The COMPILE statement must be within the set of OCL statements that apply to the compilation. The COMPILE statement must follow the LOAD or CALL statement and precede the RUN statement.

Format

// COMPILE parameters

Contents

All the parameters are keyword parameters. The keywords are:

SOURCE: The SOURCE parameter tells the system the name of the source program. The keyword SOURCE must be followed by the name of the source program on disk. The name is the name by which the source program is identified on disk in the source library. You can place source programs in a source library by using the library maintenance program. The program name you supply in the library maintenance control statements is the name used to identify the source program in the library. (For more information, see *Library Maintenance Program* in Part 4 of this manual.)

If the SOURCE parameter is not used, the source program is assumed to be in the job stream following the RUN statement.

The SOURCE parameter must be accompanied by the UNIT parameter.

UNIT: The UNIT parameter is used only when the SOURCE parameter is used.

The UNIT parameter is a code indicating the simulation area that contains the source program. Possible codes are R1, F1, R2, F2.

OBJECT: The OBJECT parameter tells the system where to place the object program. The OBJECT parameter may be specified without the SOURCE and UNIT parameters. The codes used to indicate the simulation area on which the object program is to be placed are R1, F1, R2, F2.

Note: If the OBJECT parameter is omitted, the object program is placed on the same simulation area the compiler was loaded from. The OBJECT parameter does not affect the placement of FORTRAN or COBOL object programs.

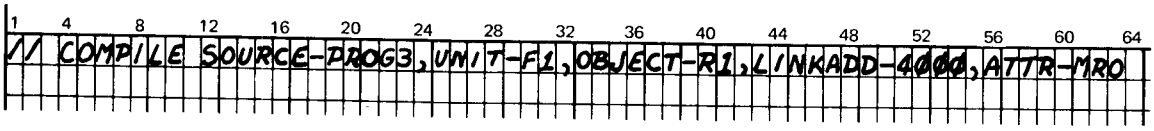
LINKADD: This parameter is the link-edit start address. The System/3 compilers (RPG II, COBOL, FORTRAN) use the overlay linkage editor to generate object modules. Normally the starting addresses for these object modules are predetermined. However, for some applications, such as executing object programs under control of CCP, it is necessary to alter the starting addresses. The LINKADD parameter can be used for this purpose (allowable entries are hex addresses 4000 or 8000).

For execution under CCP, LINKADD-8000 must be used for RPG II object programs and should be used for COBOL and FORTRAN object programs. For execution under other than CCP, LINKADD-4000 must be used for RPG II object programs (or the parameter must not be specified). For COBOL and FORTRAN object programs, LINKADD-4000 should be specified (or the parameter should not be specified).

ATTR: The ATTR-MRO parameter tells the overlay linkage editor to link-edit an object module for the REMAP technique of memory resident overlays. The ATTR-MOV parameter tells the overlay linkage editor to link-edit an object module for the MOVE technique of memory resident overlays. For additional information about the memory resident overlays, refer to the *IBM System/3 Overlay Linkage Editor Reference Manual,* GC21-7561.

Note: The ATTR parameter should be used only with an O module. (MRO and MOV program attributes will not be attached to an R module.)

The following sample COMPILE statement tells the system that the source program with the name PROG3 is located on the simulation area assigned to F1.



The OBJECT-R1 parameter tells the system to place the object program on the simulation area assigned to R1. The LINKADD-4000 parameter tells the linkage editor to link-edit the program to start at address hex 4000. The ATTR-MRO parameter tells the linkage editor to link-edit an object module for the REMAP technique of memory resident overlays.

Spooling Considerations

None

DATE Statement

Function

The DATE statement changes the system date or one of the three partition dates.

Placement

System Date

To set the system date, the DATE statement is entered during IPL before any JOB, CALL, or LOAD statements.

Partition Date

The partition date can be changed by use of the DATE statement, as follows:

1. *Job Date:* If the DATE statement is entered after the JOB statement and before a LOAD statement, that date will remain in effect for the remainder of that job. It is restored to the current system date for the next job. The DATE statement need not precede the first step; it may be placed prior to any step's LOAD statement and will be in effect from that step to the end of that job.

2. *Step Date:* If the DATE statement is entered after a LOAD or CALL statement and before the RUN statement for that step (that is, if the DATE statement is entered within a step), the partition date is changed for the duration of that step. It is restored to the current partition date for the next step.

Use of the DATE statement prior to a LOAD statement ensures that the same date is used for each step in the job. Also, once a DATE statement is entered in this manner, a subsequent DATE statement may be entered for the job, but only if it occurs prior to a LOAD statement.

Format

```
// DATE date
```

Contents

The system date can be in either of two formats: month-day-year (mmddyy) or day-month-year (ddmmyy). You must specify the format during system generation.

Example

The date can be written with or without punctuation. For example, July 25, 1993, could be specified in any one of the following ways:

```
07-25-93        250793

25-07-93        7-25-93

072593          25-7-93
```

Any characters except commas, apostrophes, numbers and blanks can be used as punctuation.

Spooling Considerations

When input spooling is present on the system, the date should be entered during IPL. This entry can be made via a command or in response to the IPL prompt for DATE. If the date is not entered during IPL, the first job to execute must have a DATE statement preceding the first JOB statement.

```
//A JOB   //B JOB   //C JOB
// LOAD   // DATE   // DATE
// DATE   // LOAD   // LOAD
// RUN    // RUN    // RUN
// LOAD   // LOAD   // LOAD
// RUN    // RUN    // DATE
/.        /.        // RUN
                    /.
```

In job A, the partition date is changed only for the first step of the job. For the second step, the date is restored to the system date.

In job B, the partition date is changed for all steps in the job. The date is restored to the system date for the next job.

In job C, the partition date is changed for all steps in the job. The DATE statement in the second step is not allowed and will cause a message to be issued.

FILE Statement (Single Volume Disk Files)

Function

The FILE statement supplies the system with information about disk files. The system uses this information to read records from and write records on disk.

Placement

You must supply a FILE statement for each of the new disk files that your programs create, and for each of the existing disk files that your programs use. The maximum number of files allowed is explained under *Scheduler Work Area* in Part 2 of this manual. The FILE statement must follow the LOAD or CALL statement and precede the RUN statement.

Format

// FILE parameters

Contents

All of the parameters are keyword parameters, as follows (keywords are in capital letters):

NAME-filename (in program)

UNIT-5444 unit code or main data area code

PACK-name

LABEL-filename (on disk)

RECORDS-number or TRACKS-number

LOCATION- { track number (simulation area only) / cylinder number / cylinder number/track number } Main data area only

RETAIN-code

DATE-date

VERIFY-code

SHARE-code

The NAME, PACK, and UNIT parameters are always required. The others are required only under certain conditions.

NAME: The NAME parameter is always needed. It tells the system the name that your program uses to refer to the file. The NAME parameter must be placed on the first record or line if two or more records or lines are used for the FILE statement. (See *General Coding Rules* for rules on continuation.)

Programs requiring specific file names for disk files are as follows:

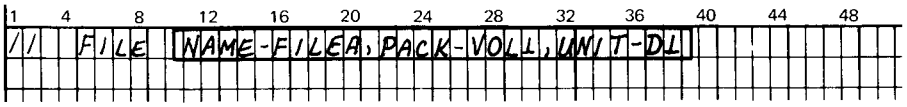
Program	File	Name	
Copy/Dump	Input	COPYIN	
	Output	COPYO	
	Output	COPYP (Optional)	
	Work	$INDEX45[1] (Optional)	
	Work	$INDEX40[3] (Optional)	
Disk Sort	Input	INPUT or INPUT1	
		INPUT2 through INPUT8	
	Work	WORK (Optional)	
	Output	OUTPUT	
CCP/Disk Sort	Input	$SOURCE	
	Work	$WORK	
	Input	INPUT or INPUT1	Names required only for generation (compile)
		INPUT2 through INPUT8	
	Work	WORK	
	Output	OUTPUT	
Dump/Restore	Input	BACKUP	
	Output	BACKUP	
Spool File Copy	Output	PRINTQ[2] (Optional)	
	Output	PUNCHQ[2] (Optional)	
	Input	READERQ[2] (Optional)	
	Output	READERQ[2] (Optional)	
	Input	CONTROL[2] (Optional)	
	Output	DISPLAYQ[2] (Optional)	
	Input	RESTORE[2] (Optional)	
	Input/update	AUTHORIZ (Optional)	
Assembler	Input	$SOURCE	
	Output	$WORK	
	Work	$WORK2	
COBOL Compiler	Input	$SOURCE	
	Work	$WORK	
	Work	$WORKX	
FORTRAN Compiler	Input	$SOURCE	
	Work	$WORK	
System History Area Display	Output	$HISTORY	
System History Area Copy	Output	$SHAFILE	
RPG II Compiler	Input	$SOURCE	
	Work	$WORK	
RPG II Auto Report	Input	$SOURCE	
	Work	$WORK	

[1] If you supply a LABEL parameter in the FILE statement, it must be LABEL-$INDEX45.
[2] The file name can be replaced by the name specified on a control statement parameter.
[3] If you supply a LABEL parameter in the FILE statement, it must be LABEL-$INDEX40.

Program	File	Name	
Macro Processor	Output	$SOURCE	
Overlay Linkage Editor	Input Work	$SOURCE $WORK	Optional – either both present or both absent
Spool (See Note)	Work	$SPOOL	
Any program adding to large indexed files or loading a large unordered indexed file.	Work	$INDEX45[1] (Optional) (for main data area file) or $INDEX40[2] (Optional) (for main data area file)	

Note: The $SPOOL file is internally generated by spooling and cannot be referenced on a FILE statement.

The keyword, NAME, must be followed by the filename used by the program. The filename can be any combination of characters except commas, apostrophes, or blanks. The first character must be alphabetic. The number of characters must not exceed 8. The following example shows how the NAME parameter for a file named FILEA would be coded:



UNIT: The UNIT parameter is always needed. It tells the system the simulation area or main data area that contains or will contain the file. The keyword, UNIT, must be followed by a code that indicates the area. Possible codes are R1, F1, R2, F2, D1, D2, D3 or D31, D32, D33, D34, D4 or D41, D42, D43, D44.

The previous example shows how the UNIT parameter for a file located on the main data area D1 would be coded.

PACK: The PACK parameter is always needed. It tells the system the name of the area that contains or will contain the file. The system checks this name to ensure that the proper area is being used. (For information about how an area is given a name, see *Disk Initialization Program* or *Simulation Area Program* in Part 4 of this manual.)

The keyword, PACK, must be followed by the name of the area. The example under NAME shows how the PACK parameter for a file on an area named VOL1 would be coded.

[1] If you supply a LABEL parameter in the FILE statement, it must be LABEL-$INDEX45.
[2] If you supply a LABEL parameter in the FILE statement, it must be LABEL-$INDEX40.

LABEL: The LABEL parameter tells the system the name by which your file is identified or will be identified.

If the file is being created, the name you supply in the LABEL parameter is used to identify the file. If you omit the LABEL parameter from a disk FILE statement, the name from the NAME parameter is used.

When the name your program uses to refer to an existing disk file differs from the name by which the file is identified, you must supply a LABEL parameter.

The keyword, LABEL, must be followed by the name of the file. The name can be any combination of characters except commas, apostrophes, or blanks. The first character must be alphabetic. The number of characters must not exceed 8. The LABEL parameter for a file named PAYROLL is coded in the following example:

```
1   4       8       12      16      20      24      28      32      36      40      44      48      52
//  FILE NAME-OFILE2,LABEL-PAYROLL,UNIT-D2,PACK-VOL1
```

TRACKS or RECORDS: The TRACKS or RECORDS parameter is needed for files that are being created. The parameter tells the system the amount of space needed on disk for the file.

If you use the TRACKS keyword, you specify the number of disk tracks needed for the file.

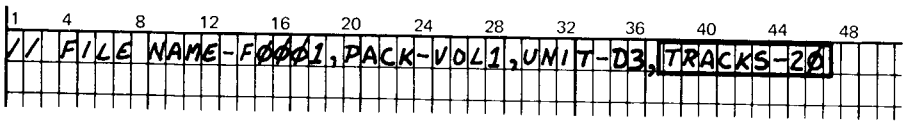
If you use the RECORDS keyword, you specify the approximate number of records for the file. The total space allocated will be rounded up to full tracks, allowing adequate space to accommodate at least the number of records indicated. This means the file could hold more records than specified on the RECORDS keyword, allowing you to add more records to the file. Therefore, when using the copy/dump program ($COPY) to copy the file to another disk, you may have to specify more records than were specified in the RECORDS keyword when the file was created.

Either TRACKS or RECORDS can appear in the FILE statement, but not both. The keyword must be followed by a number indicating the amount of space needed.

When loading a new file at the same location as an existing file, using the LOCATION parameter, you must specify the same parameter (TRACKS or RECORDS) that was used when the existing file was created. (The existing file must be a temporary file.)

Several versions of a file can be created on the same disk and be given the same name. If the TRACKS or RECORDS parameter you are using in creating a file is the same as the TRACKS or RECORDS specified for an existing file, you must specify LOCATION. You can reference each of these files by its name and date, or by its name and location on disk. Both date and location must be unique for each version. (See *Example 2, Example 4,* and *File Processing Considerations.*)

If TRACKS is used, the number must be within the range of 1–398 for a simulation area; 1–3320 for a main data area on a 3340, and 1–3720 for a main data area on a 3344. The following example shows how the TRACKS parameter for a file requiring 20 tracks is coded:



If RECORDS is used, the number can be up to six digits long. The RECORDS parameter for a file containing 250 records is coded as follows:

LOCATION: LOCATION is required when you create a file with the same LABEL (the date must be unique) and the same size as one that already exists (LOCATION is not required if sizes differ), load to an existing file, and load an offline multivolume file to volumes that contain other files.

The LOCATION parameter can be used to specify the first track of a new file. It may also be used when a file is referenced, for a more specific identification check, and for identifying one of several files having the same name and same size.

If you are creating a file, this parameter tells the system the number of the track on which the file is to begin. If you omit the parameter, the track is chosen for you. The system places the file in the smallest available space on the pack, leaving as few empty spaces as possible.

If you are referencing a file, the parameter tells the system the number of the track on which the file begins. In this case, the system uses the track number to distinguish one file from another.

For a simulation area, the LOCATION format is:

LOCATION-track number

Track number must be between 8 and 405.
Tracks 0–7 are reserved for the system.

For a main data area, the LOCATION format is:

Slash is needed to separate cylinder number and track number (when both are specified)

LOCATION-cylinder number/track number

Track number must be between 0 and 19. Track number 0 is assumed if track number is not specified.

Cylinder number must be between 1 and 166 (main data area on 3340) or 1 and 186 (main data area on a 3344). Cylinder 0 is reserved for the system.

RETAIN: The RETAIN keyword must be followed by a code that indicates the classifications of the file. The codes are:

Code	Meaning
S	Scratch file
T	Temporary file
P	Permanent file

A scratch file is used only once in a program and cannot be retrieved after the program has ended. To remove a permanent file you must use the file delete ($DELET) system service program. You can remove a temporary file by using the file delete ($DELET) system service program or by using a RETAIN-S parameter. A temporary file can be changed to a permanent file only if the file name is changed or copied as a permanent file.

A temporary file is usually used more than once. The area containing a temporary file can be given to another file only under one of the following conditions:

- A FILE statement containing the RETAIN-S parameter is supplied for the temporary file and the file referenced (opened and closed) by the user program. This statement removes the file from the VTOC (volume table of contents) when the program with the FILE statement goes to end of job. The user must be aware of any external indicators that condition the use of that file. More information about deleting files is provided under *General Results When the 2 or 3 Option for a Message is Selected* in the *IBM System/3 Model 15 System Messages,* GC21-5076.

- Another file with the same LABEL name is loaded into the exact area occupied by the temporary file, but this only changes the data. Space and location parameters are required. You must specify the same parameter (TRACKS or RECORDS) that was used when the existing file was created. For example, if the TRACKS parameter was specified when the FILE was created, you must use the TRACKS parameter when reloading the same location on the pack.

- The file delete program is used to delete the file.

The area containing a permanent file cannot be used for any other file until the file delete program has deleted the permanent file.

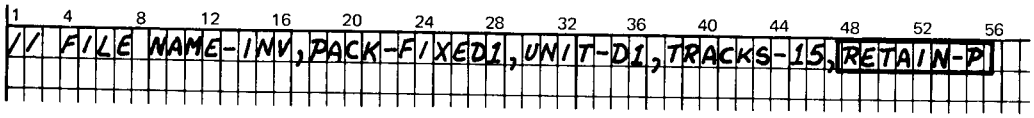
A disk file is classified as scratch, temporary, or permanent when it is created. If the RETAIN parameter is omitted from the FILE statement when the file is created, the file is assumed to be a temporary file. You may omit the RETAIN parameter when accessing an existing file.

Notes:

1. The *output* file will be scratched (deleted) if all three of the following conditions exist at end of job step:
   a. A pack containing an input file is not online at the start of the job (deferred mount).
   b. The output file is to be written over the input file (load to old).
   c. RETAIN-S is used on the FILE statement for the *input* file.

   To prevent the deletion of the *output* file, you should use RETAIN-T for the *input file.*
2. You should reload an existing temporary file (load to old) with files of like attributes. If an existing indexed file is reloaded with a sequential file, the new data will overlay only the data portion of the indexed file. The index portion of the file will remain intact but will not be usable.

The RETAIN parameter for a permanent file is coded as follows:

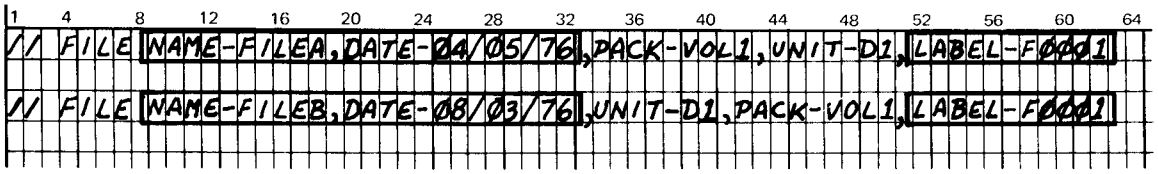


DATE: The DATE parameter tells the system the creation date of an input file. It is used to ensure that the proper version of the file is used. The date specified is compared with the creation date contained in the file label. No comparison is done when DATE is not specified.

For output files, the partition date is always used as the creation date. If the DATE parameter is specified for an output file, the system compares the specified date with the creation date of the existing file. If no file exists, or if the dates do not agree, the system halts. (See *Interval Timer* for information on the effect of the interval timer on date.)

The date may be coded in one of two formats: month-day-year (mmddyy), or day-month-year (ddmmyy). The format must match the format of the system date chosen during system generation. The date may be coded with or without punctuation. Blanks, commas, numbers, or apostrophes are not allowed as punctuation. Leading zeros in month and day may be omitted if punctuation is used.

To illustrate this parameter, assume that two versions of a file are written on the same main data area. In the next example are the NAME, LABEL, and DATE parameters for two versions of a file on the same main data area, one written on April 5, 1976, the other on August 3, 1976. Both files have the same label: F0001.



VERIFY: The VERIFY parameter is used to specify verification of disk write operations for this file in this step (main data area only). If VERIFY-YES is specified, verification takes place. If VERIFY-NO is specified, write verification is bypassed. If VERIFY is not specified, VERIFY-YES is assumed unless RETAIN-S is coded, in which case VERIFY-NO is assumed. Verification is always done when a simulation area is accessed.

SHARE: The SHARE parameter is used to allow or disallow file sharing between partitions if the access methods are compatible. If SHARE-YES is specified, file sharing is allowed between partitions (offline multivolume files cannot be shared); SHARE-NO does not allow file sharing between partitions. If RETAIN-S is specified on the FILE statement, file sharing is not allowed. If this parameter is omitted, SHARE-YES is assumed. For additional information about file sharing, refer to *File Sharing* in Part 2 of this manual.

Examples

The following are examples of FILE statements. In each example, the file is described first, then the corresponding FILE statement is shown.

Example 1: Suppose that each week you create a disk file that contains the records for the transactions you had made that week. Assume the following facts about that file:

- The name your program uses to refer to the file is TRANS, which is also the name you want to use to identify the file on disk.
- You are placing the file in a main data area named VOL03.
- You intend to mount the data module on drive 2.
- You want to save the file for use at the end of the month.
- The file contains 225 records.
- You are letting the system choose the area that will contain the file.

The following example shows how the FILE statement for the preceding file is coded:

Example 2: Suppose you had created, on the same data module (VOL03), four versions of the transaction file described in the preceding example—one for each of the weeks in February 1976. Assume the following:

- You had created the files on the following days: 2/6/76, 2/13/76, 2/20/76, and 2/27/76 (these were the system dates used for each of the files).
- You want to reference the third file (the one created 2/20/76).
- You intend to mount the data module on drive 2.

The FILE statement you would need is:

```
1   4   8   12  16  20  24  28  32  36  40  44  48  52
// FILE NAME-TRANS,DATE-02/20/76,PACK-VOL03,UNIT-D2
```

Example 3: Suppose that at the end of the month you combine the files referred to in example 2, for use in preparing your monthly bills. Further assume the following:

- Your program uses the name TRANS to refer to the file, but you want to use the name BILLING to identify the file on disk.
- You are expressing the amount of disk space as the number of tracks required to contain the file (assume the number is 15), and you want the file to begin on cylinder 8, track 0.
- You are placing the file in a main data area named VOL03.
- You intend to mount the data module on drive 2.

The following example shows the FILE statement you would use for this file.

```
1   4   8   12  16  20  24  28  32  36  40  44  48
// FILE NAME-TRANS,LABEL-BILLING,
//      UNIT-D2,PACK-VOL03,
//      TRACKS-15,LOCATION-8,
//      RETAIN-T
```

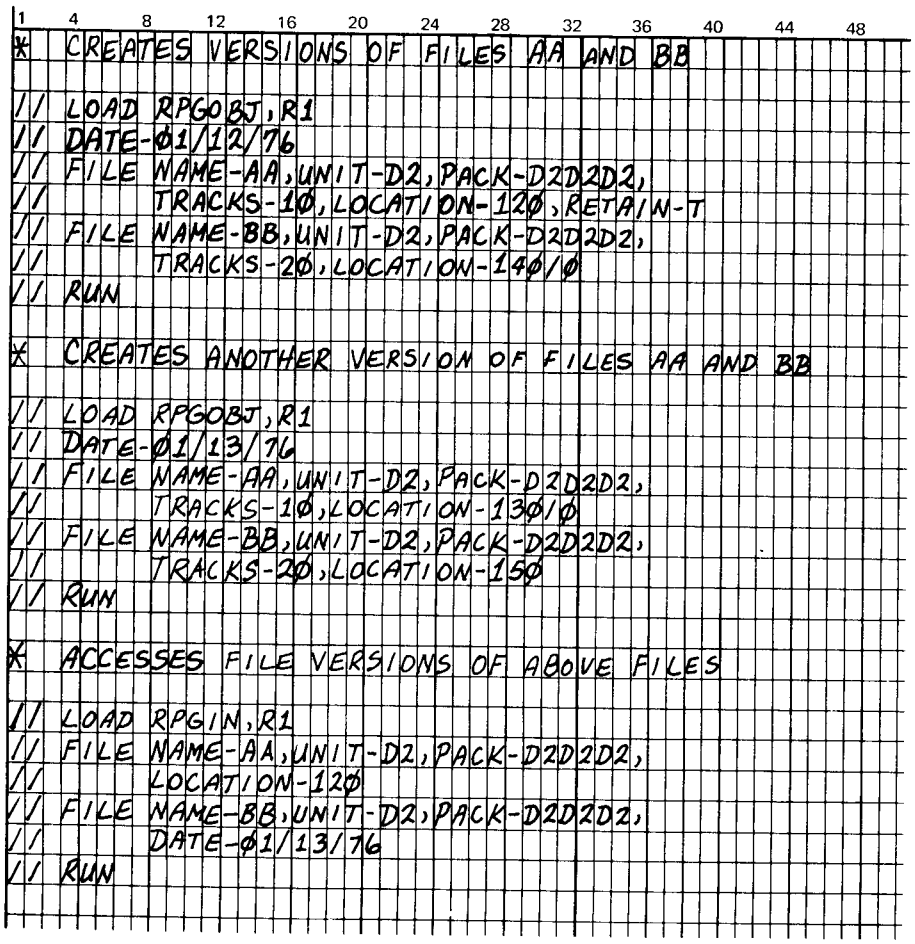
Example 4: Suppose you want to create two versions of two files on disk and later to access one version of each file. Further assume the following:

- The names your program uses to refer to the files are AA and BB, which are also the names you want to use to identify the files on disk.
- Files AA and BB are being placed on a data module on drive 2 named D2D2D2.
- One version of each file is created on 1/12/76 and 1/13/76.
- Disk space and location for the files are:

File	Version	Tracks	Location
AA	1/12/76	10	120/0
	1/13/76	10	130/0
BB	1/12/76	20	140/0
	1/13/76	20	150/0

- You want to access file AA, version 1/12/76, and file BB, version 1/13/76.

The following OCL statements are needed to create the above versions of files AA and BB and to access a version of each file.



File Processing Considerations

LOCATION and space (TRACKS or RECORDS) must be specified when you are reloading an existing temporary file. You must specify the same parameter (TRACKS or RECORDS) that you used when the existing file was created.

If you are referencing a file by the DATE parameter and space is given, the space must be equal to the space given when that file was created.

If you are accessing a file by the LOCATION parameter and space is given, the space must be equal to the space given when that file was created.

You can create several versions of a file with a program by changing the locations of the files and using different partition dates.

You can create different versions of a file without LOCATION if the space parameters as well as the partition dates are different.

The DATE parameter is allowed only for accessing existing files.

Whenever a load is performed to an existing file (load to old), the partition date replaces the previous date for that file.

The only file that can be reloaded is a file that has a RETAIN-T classification.

When a scratch file is created, it is not entered in the volume table of contents (VTOC). After the job step that created the file is executed, the file cannot be accessed.

Spooling Considerations

None

FILE Statement (Multivolume Disk Files)

Function

The FILE statement supplies the system with information about disk files. The system uses this information to read records from and write records on disk.

Placement

You must supply a FILE statement for each of the new disk files that your programs create, and for each of the existing disk files that your programs use. The FILE statement must follow the LOAD or CALL statement and precede the RUN statement.

Format

// FILE parameters

Contents

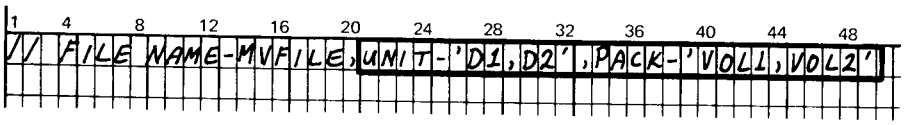
The FILE statement for multivolume disk files requires special considerations when you define and code these keyword parameters: PACK, UNIT, TRACKS or RECORDS, HIKEY, and LOCATION. The maximum number of multivolume files allowed is explained under *Scheduler Work Area* in Part 2 of this manual. These considerations are necessary for the following reasons:

- When processing disk files contained on more than a single volume, the system requires information about each volume to perform all the necessary protection and checking functions.
- Additional information is needed to determine and check the sequence in which the volumes are processed and when they are to be mounted on the disk drives.

The rules for coding a list of data or codes after a keyword are as follows:

- The list must be enclosed by apostrophes.
- The items in the list must be separated by commas. No blanks are allowed within or between items (HIKEY can contain blanks).

The following example shows lists in parameters. The file is an online multivolume file (number of units = number of volumes).



The PACK, LOCATION, TRACKS or RECORDS, and HIKEY parameters require lists. The UNIT parameter may require a list. The considerations for using the lists in these parameters are included in the following parameter discussions. (Parameters not mentioned here are used as explained under the *FILE Statement [Single Volume Disk Files].*)

PACK: The names of the volumes that contain or will contain the multivolume file must follow the keyword PACK. (PACK names must be unique for proper functioning.)

When a multivolume file is created, the system writes a sequence number on the disks to indicate the order of the volumes. The volumes are numbered in the order in which you list their names in the PACK parameter.

When a multivolume file is processed, the system provides two checks to ensure that the volumes are used in the proper order:

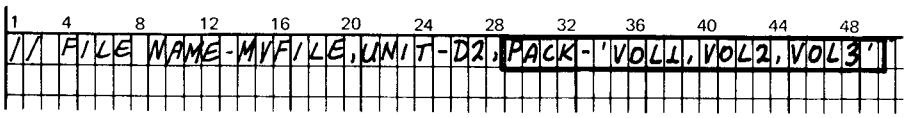
- It checks to ensure that the volumes are used in the order that their names are listed in the PACK parameter.
- It checks the sequence numbers of the volumes used to ensure that they are consecutive and in ascending order (01, 02, and so on).

The system stops when it detects a volume that is out of sequence. The operator can do one of three things:

- Mount the proper volume (if dismount is allowed) and restart the system.
- Restart the system and process the volume that is mounted if the sequence is ascending (for indexed files processed offline, consecutive input, and consecutive update processing).
- End the program.

Consecutive input or update sequence numbers do not exist if the file was not created as a multivolume file. If a file is created as multivolume and the sequence is ascending but not consecutive, a diagnostic halt is given.

The following is an example of the PACK parameter for an offline multivolume file that is contained on three volumes, named VOL1, VOL2, and VOL3:



UNIT: The keyword UNIT must be followed by a code or codes indicating the location on the disk unit that contains or will contain the file. No UNIT parameter may be repeated. Possible codes for offline multivolume files are D1 and/or D2. (D1 is a possible code only if the IPL was performed from drive 3). Possible codes for online multivolume files are D1, D2, D3 or D31, D32, D33, D34, D4 or D41, D42, D43, and D44.

The order of codes in the UNIT parameter must correspond to the order of names in the PACK parameter.

When the system is processing offline multivolume files and more than one unit is specified, the unit parameters must be in ascending sequence.

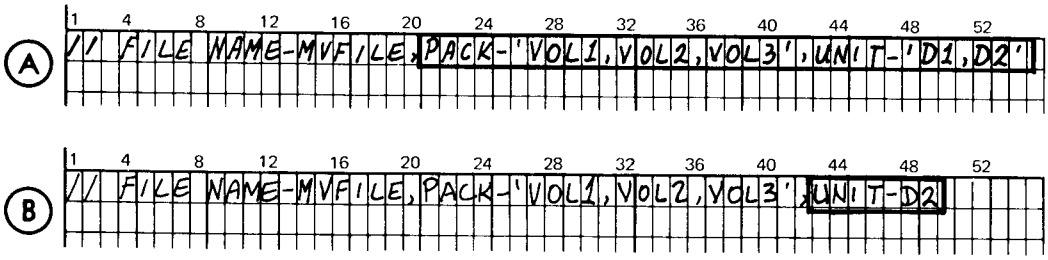
When you are creating or processing a sequential or indexed file, you can use the same drive for more than one of the volumes; however, the volumes must then all be removable. If you do use the same drive, you must not repeat the code for the drive in the UNIT parameter. When the number of codes in the UNIT parameter is less than the number of names in the PACK parameter, the system uses the codes alternately.

If you specify a volume on drive 3 or drive 4, the file must be online multivolume.

Assume that your program processes an offline file consecutively. Further assume the following:

- The disks containing the file are named VOL1, VOL2, and VOL3, respectively.
- You intend to mount VOL1 and VOL3 on 3340 drive 1, and VOL2 on 3340 drive 2. (The IPL must be performed from drive 3.)

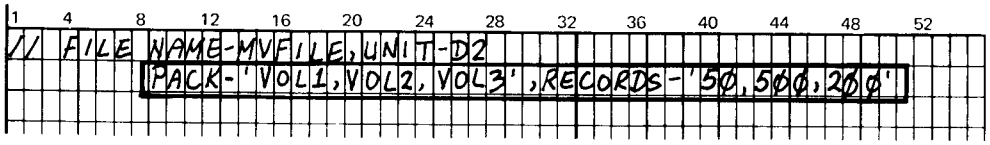
In the following, (A) shows the PACK and UNIT parameters for the file. If all three volumes were used on 3340 drive 2, the UNIT parameter in (B) would have been used.



TRACKS or RECORDS: These keywords, TRACKS or RECORDS, must be followed by numbers that indicate the amount of space needed on each of the volumes that will contain the multivolume file. TRACKS or RECORDS must be specified. Any multivolume file load requires a TRACKS or RECORDS parameter whether the file previously existed or not. The order of these numbers must correspond to the order of the names in the PACK parameter. For example, assume the following:

- The program is creating a sequential (offline) file on three volumes: VOL1, VOL2, and VOL3.
- The first 50 records are to be placed on VOL1, the next 500 on VOL2, and the last 200 on VOL3.

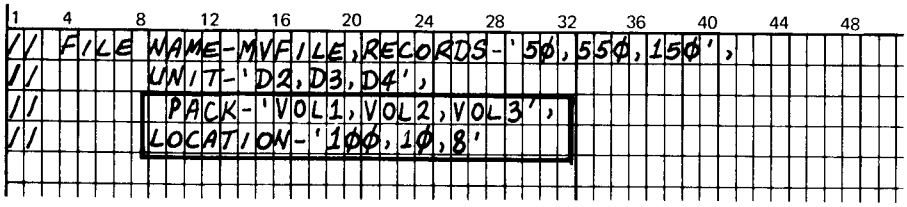
The PACK and RECORDS parameters for the file are:



LOCATION: The keyword LOCATION must be followed by the numbers of the tracks on which the file is to begin on each of the volumes you use for the file. The order of the numbers must correspond to the order of the names in the PACK parameter. For example, assume the following:

- The volumes containing the file are VOL1, VOL2, and VOL3.
- The file is to begin on cyl 100 in VOL1, cyl 10 in VOL2, and cyl 8 in VOL3.

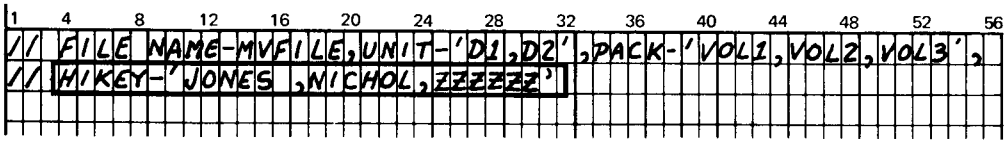
The PACK and LOCATION parameters for the file are shown in the following example.



If you omit the LOCATION parameter, the system chooses the beginning track on each of the volumes. When an offline multivolume file is created, offline volumes cannot contain files if the LOCATION parameter is not specified. If LOCATION is specified for one volume, it must be specified for all volumes. If the multivolume file exists, LOCATION given for all volumes must be identical to the LOCATION parameters specified when the file was created.

RETAIN: RETAIN-S must not be specified unless the file is online multivolume.

HIKEY: The HIKEY parameter is used only for multivolume indexed files. HIKEY limits the highest key field that can be put on each volume of a multivolume file. The following example contains an example of a HIKEY parameter list. In this case, the three volumes contain lists of names. The highest keyfield allowed on the first volume is JONES. This means that all the records beginning with A and including JONES will be processed on this volume. Since HIKEY parameters must be in ascending order, the next volume will contain all of the records with names following JONES and including NICHOL. The last volume will contain all the records with names that come after NICHOL.



OCL considerations for the HIKEY parameter are:

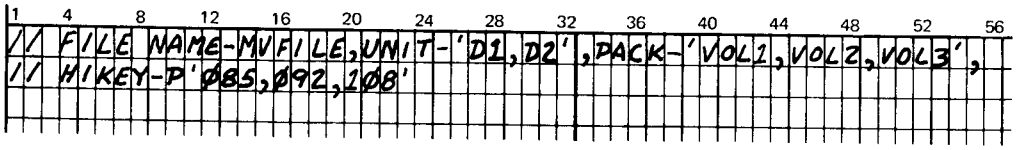
- All characters except commas are valid.
- The list of HIKEY parameters must begin and end with an apostrophe even if only one parameter is specified. A single apostrophe in a key field must be written as a double apostrophe in the HIKEY parameter.
- For each PACK parameter specified, there must be a corresponding HIKEY key field parameter for that pack.

- The HIKEY fields must be equal in length and must be specified in ascending order.
- The maximum length of a HIKEY field is 29 characters.
- The HIKEY fields must be the same length as the keys on file.
- Continuation of HIKEY sublists must begin in column 4 of the continuation record following the // blank.
- Comments must not follow the last comma on a FILE statement when the last parameter is an incomplete HIKEY sublist.

Packed HIKEY: The packed HIKEY parameter has all the OCL considerations for HIKEY, including the following restrictions:

- The first character following the HIKEY keyword and hyphen (HIKEY-) must be a P to indicate packed HIKEY.
- All characters in the packed HIKEY must be zoned numerics (0–9).
- The number of digits in each packed key must be the same.
- The number of zoned numeric characters per packed HIKEY must not exceed 15, since the maximum packed key field length is 8.

The following example shows a packed HIKEY parameter. In the example the key field length of MVFILE is 2. The HIKEYs are X'085F', X'092F', and X'108F' for VOL1, VOL2, and VOL3, respectively. The first two packed keys required a leading zero to make the lengths consistent.



Spooling Considerations

None

FILE Statement (Single Volume Tape Files)

Function

The FILE statement supplies the system with information about tape files. The system uses this information to read records from and write records to tape.

Placement

You must supply a FILE statement for each new tape file that your program creates and for each existing tape file that your program uses. The maximum number of files allowed is explained under *Scheduler Work Area* in Part 2 of this manual.

Format

// FILE parameters

Contents

All parameters are keyword parameters. The parameters are as follows (keywords are in capital letters):

NAME-filename (in program)

UNIT-code

REEL- { name / NL / NS / BLP }

LABEL- { filename (on tape) / 'character string' }

DATE-date

RETAIN-code

BLKL-block length

RECL-record length

RECFM- { F / V / D / FB / VB / DB }

END- { LEAVE / UNLOAD / REWIND }

DENSITY- { 200 / 556 / 800 / 1600 }

ASCII- { YES / NO }

DEFER- { YES / NO }

CONVERT- { OFF / ON }

TRANSLATE { ON / OFF }

PARITY- { EVEN / ODD }

SEQNUM- { X / Sequence number }

The NAME and UNIT parameters are always required. The others are required only under certain conditions.

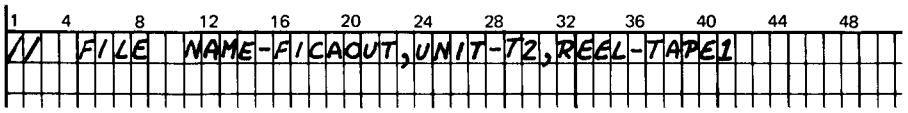
NAME: The NAME parameter is always required. It tells the system the name that your program uses to refer to the file. The NAME parameter must be placed on the first card or line if two or more cards or lines are used for the FILE statement. (See *General Coding Rules* for rules on continuation.)

Programs requiring specific filenames for tape files are as follows:

Program	File	Name
Tape Sort	Input	INPUT
	Output	OUTPUT
	Work	WORK1
		WORK2
		WORK3
		WORK4 (optional)
Copy/Dump	Input	COPYIN
	Output	COPYO
Disk Sort	Input	INPUT or INPUT1
		INPUT2 through INPUT8
	Output	OUTPUT
Dump/Restore	Input	BACKUP
	Output	BACKUP
$HIST	Output	$HISTORY
$FCOMP (tape support only)	Input	BACKUP
	Output	BACKUP

Program	File	Name
Spool File Copy	Input	$SPOOL (optional)
	Output	$SPOOL (optional)
	Input	READERQ[1] (optional)
	Input	RESTORE[1] (optional)
	Output	DISPLAYQ[1] (optional)
	Output	PRINTQ[1] (optional)
	Output	PUNCHQ[1] (optional)
	Output	READERQ[1] (optional)

The keyword NAME must be followed by the filename used by the program. The first character of the NAME must be alphabetic. The remaining characters may be any combination of characters except commas, apostrophes, or blanks. The number of characters must not exceed 8. The following example shows how the NAME parameter for a file named FICAOUT would be coded:



UNIT: The UNIT parameter is always required. It tells the system the tape unit that contains or will contain the file. The keyword UNIT must be followed by a code that indicates the unit. The codes are as follows:

Code	Meaning
T1	Tape unit 1
T2	Tape unit 2
T3	Tape unit 3
T4	Tape unit 4

The previous example shows how the UNIT parameter would be coded for a file that resides on tape unit 2.

[1]The file name can be replaced by the name specified on a control statement parameter.

REEL: The REEL parameter is required for tape input files and optional for output files. It identifies the tape that contains or will contain the file. The system uses this parameter to ensure that the correct tape is being used. (For information about how a tape is initialized and identified, see *Tape Initialization Program – $TINIT* in Part 4 of this manual.) The keyword REEL must be followed by one of the following codes:

REEL-nnnnnn	This format is used for labeled tape files. You identify the volume by coding a maximum of 6 characters, excluding commas, apostrophes, and blanks. NS, NL, and BLP have special meanings and may not be used as the name of the reel.
REEL-NL	This coding indicates a tape file without a label. The first record of an unlabeled tape must not be an 80-byte record with VOL1 as the first 4 characters.
REEL-NS	This coding indicates an input tape file with a non-standard label. These labels do not adhere to the IBM Tape Label Standard. The first record of a nonstandard labeled tape must not be an 80-byte record with VOL1 as the first 4 characters. REEL-NS is invalid for output files.
REEL-BLP	This coding indicates that label processing of standard labeled input tapes should be bypassed.

If the REEL parameter is not specified for an output file, the system assumes that the output tape contains standard labels. If REEL-NS or REEL-NL is used, the LABEL, DATE, and RETAIN parameters may not be entered. REEL-BLP may not be specified for an output tape.

Note: User labels are file labels that follow standard header and trailer label conventions (ANSI or IBM). They are a variation of standard labels with a partially fixed format. These labels are sometimes provided by other systems. User labels are not checked by Model 15 tape data management and may not be written as part of the label group.

The example under NAME shows how the REEL parameter would be coded for a file on a tape named TAPE1.

LABEL: The LABEL parameter tells the system the name (label) of the tape file as it exists in the header label.

For file creation, the name you supply in the LABEL parameter is used in the header label. If you omit the LABEL parameter, the name from the NAME parameter is used unless REEL-NS, REEL-NL, or REEL-BLP is also specified. Up to 8 characters may be supplied in the LABEL parameter.

For existing files, you must supply the LABEL parameter if the name in the tape label is different from the name your program uses to refer to the file (the NAME parameter). If the header label contains a name longer than 8 characters, only the first 8 characters are recognized by the system for comparison.

The LABEL parameter may not be used with the parameters REEL-NS, REEL-NL, or REEL-BLP. The LABEL parameter can be coded as follows: LABEL-name.

The name entry must begin with an alphabetic character and the remaining characters must not be commas, apostrophes, or blanks.

A label may also be identified by special characters. The character string must be enclosed in apostrophes, may not contain commas, and may not be longer than 8 characters; for example, LABEL-'character string'. If an apostrophe is used as a character, it must be coded as 2 apostrophes.

DATE: The DATE parameter tells the system the creation date of an input file. It is used to ensure that the proper version of the file is used. The date specified is compared with the creation date contained in the file label. No comparison is done when DATE is not specified.

For output files, the partition date is always used as the creation date. If the DATE parameter is specified for an output file, the system compares the specified date with the creation date of the file already on the tape. If no file exists on the tape, or a file with a different label exists, or the dates do not agree, the system halts. (See *Interval Timer* or *DATE Statement* for more information about the effect of the interval timer on date.)

The date may be coded in one of two formats: month-day-year (mmddyy) or day-month-year (ddmmyy). The format must match the format of the system date chosen during system generation. The date may be coded with or without punctuation. Blanks, commas, numbers, or apostrophes are not allowed as punctuation. Leading zeros in month and day may be omitted if punctuation is used.

The DATE parameter may not be specified with REEL-NS, REEL-NL, or REEL-BLP.

RETAIN: The RETAIN parameter specifies the number of days a file should be retained before it expires. This number may be from 0 to 999. After the number of days has elapsed, the file expires and the system allows the file to be written over. If the RETAIN parameter is omitted, a value of zero is assumed. A value of 999 indicates a non-expiring permanent tape file.

If an attempt is made to write over an unexpired file, the system halts, allowing the operator to cancel the job or continue. A tape containing a permanent tape file must be reinitialized before it can be used for output. The RETAIN parameter may not be used with REEL-NS, REEL-NL, or REEL-BLP.

BLKL: The BLKL (block length) parameter specifies the number of bytes in a physical block of data on tape. The minimum size fixed length block (FB) that can be specified is 18. Variable length (VB or DB) blocks are padded with hex 00 (EBCDIC) or hex 5E (ASCII) when necessary to meet the 18-byte minimum block length. The maximum size block that can be specified, regardless of record format, is 32,767. When fixed blocked (FB) records are used, block length must be an integral multiple of record length. For a file containing blocked EBCDIC variable length (VB) records, the block length must include the 4-byte block descriptor and the 4-byte record descriptor(s). For blocked ASCII variable length (DB) records, the buffer offset length and the 4-byte record descriptor(s) must be included in the block length.

RECL: The RECL (record length) parameter specifies the number of bytes in a logical tape record. The minimum record length for fixed (F) or fixed block (FB) records is 18 bytes. Unblocked variable length (V or D) records are padded with hex 00 (EBCDIC) or hex 5E (ASCII), when necessary, to meet the 18-byte minimum block length requirement. For files containing variable length (V or D) records, the record length must include the 4-byte record descriptor.

RECFM: The RECFM (record format) parameter identifies the format of the input or output file records. The parameter entries are:

F	Fixed length, unblocked records. Logical and physical records are the same size.
V	Variable length, unblocked records. Each physical record contains one logical record; the logical record can vary in length.
D	Variable length, unblocked records in the D-type ASCII format.
FB	Fixed length, blocked records. All records are of equal length and all blocks are of equal length. Each physical record contains more than one logical record.
VB	Variable length, blocked records. Each physical record contains logical records of various lengths.
DB	Variable length, blocked records in the D-type ASCII format.

END: The END parameter specifies the position of the tape after the file has been processed. The options are as follows:

LEAVE	The tape remains in the position it was in after the last record was read or written.
REWIND	The tape is rewound to the load point.
UNLOAD	The tape is rewound and unloaded for removal from the tape drive.

If the END parameter is omitted, REWIND is assumed.

DENSITY: The DENSITY parameter is used to specify the number of bpi (bits per inch) at which files are to be written or read.

The parameter must specify the density at which the tape was initialized. See *$TINIT* (tape initialization program) description in Part 4 of this manual. For 9-track tapes, this parameter affects only the density of nonlabeled output files. When standard labeled or nonstandard labeled tapes are used, the 9-track tape hardware automatically determines the density at which the tape was initialized. When a tape is initialized to 1600 bpi with standard lables, any file that is written on that tape is in 1600 bpi, regardless of the parameter specified for DENSITY. No error halts occur if the wrong 9-track density is specified. The parameter entries are:

1600	The file is to be written at 1600 bpi (valid for all 9-track tape units).
800	The file is to be written (7- or 9-track tape units) or read (7-track tape units) at 800 bpi (valid for 9-track dual density tape units or for all 7-track tape units).
556	The file is to be written or read at 556 bpi (valid for all 7-track tape units).
200	The file is to be written or read at 200 bpi (valid for all 7-track tape units).

If the DENSITY parameter is omitted, 1600 bpi is assumed on 9-track tape units, and 800 bpi is assumed on 7-track tape units.

ASCII: The ASCII parameter is used to indicate to the system when an ASCII file is being used. If ASCII files are being used, ASCII-YES must be coded. ASCII-YES is invalid for files on 7-track tape units. If this parameter is omitted or coded ASCII-NO, an EBCDIC file is assumed.

DEFER: The DEFER parameter tells the system whether the file will be mounted on a tape drive when the file is allocated and opened. If the tape volume is not online, DEFER-YES must be coded. If the parameter is omitted, DEFER-NO is assumed.

Note: For RPG II object programs, this option should be used only for files that use the same drive as a table file. All other files are allocated and opened at the beginning of the program.

Other programs (such as COBOL object programs), which do not allocate and open all files at the same time or do it conditionally by program logic, should not use the DEFER-YES option.

CONVERT: The CONVERT parameter tells the system whether the data converter will be turned on or off. This parameter is valid only for 7-track tape files. CONVERT-ON causes 7-track data to be processed in 8-bit binary form. The converter writes three main storage characters as 4 tape characters and converts the opposite way when reading. CONVERT-ON must be specified when variable length records are processed on 7-track tape files. Specifying both CONVERT-ON and TRANSLATE-ON is invalid. If this parameter is omitted, CONVERT-OFF is assumed.

TRANSLATE: The TRANSLATE parameter tells the system whether the data translator will be turned on or off. This parameter is valid only for 7-track tape files. TRANSLATE-ON causes 7-track data to be processed in 6-bit BCD form. The translator writes 8-bit EBCDIC main storage characters as 6-bit BCD tape characters and translates the opposite way when reading. Specifying both TRANSLATE-ON and CONVERT-ON is invalid. If this parameter is omitted, TRANSLATE-OFF is assumed.

Note: If CONVERT-OFF and TRANSLATE-OFF are specified, only the 6 low-order bits of the main storage character are written on the tape. When the system is reading with CONVERT-OFF and TRANSLATE-OFF, the 2 high-order bits of the main storage characters are set to zeros.

PARITY: The PARITY parameter is used to specify the parity at which tape characters will be processed. This parameter is valid only on 7-track tape files. Data conversion (CONVERT-ON) is invalid with even parity (PARITY-EVEN). If this parameter is omitted, PARITY-ODD is assumed.

Note: The following are the valid combinations for TRANSLATE, CONVERT, and PARITY parameters:

PARITY-ODD, TRANSLATE-OFF, CONVERT-OFF

PARITY-ODD, TRANSLATE-ON

PARITY-ODD, CONVERT-ON

PARITY-EVEN, TRANSLATE-OFF, CONVERT-OFF

PARITY-EVEN, TRANSLATE-ON

SEQNUM: The SEQNUM parameter is used to specify the number of the file when the reel contains more than one file (multifile volume). The number to use is the number that was assigned when the file was written. The default value is 1.

If SEQNUM-number is used with REEL-BLP, the system will search the tape for a standard label HDR1 record that contains the same file number. When the record is found, further processing of the label group is terminated. The system then positions the tape to the file data.

SEQNUM-X on the FILE statement indicates that the tape has been previously positioned to the desired file, and no positioning is done before processing. See *FILE Statement (Multifile Tape Volumes)* for more information on the uses of this keyword.

7-Track Considerations

CONVERT, TRANSLATE, PARITY and/or DENSITY must be specified for an input file if other than the default parameters were specified for output when the file was built; otherwise, tape runaway or data check occurs.

If an output file has REEL-NL on the FILE statement, the reel must have been initialized with REEL-NL by the $TINIT (tape initialization) program; otherwise, tape runaway or data check occurs.

If an output file has REEL-NL on the FILE statement and there is a file existing on the tape, tape runaway or data check will occur if TRANSLATE, CONVERT, PARITY, and/or DENSITY parameters for the new file do not match the characteristics of the old file. Reinitialize the tape using $TINIT with REEL-NL if this occurs.

Spooling Considerations

None

TAPE FILE STATEMENT SUMMARY

Req = Required
Opt = Optional (Refer to the discussion of a particular parameter to determine whether it is required for your program.)
N/A = Not Applicable
Defaults are underlined

Parameters	Applicability		
	9-Track	7-Track	Remarks
NAME-filename (in program)	Req	Req	1-8 characters
UNIT- T1 T2 T3 T4	Req	Req	
REEL-name NL NS BLP	Req*	Req*	NL, NS, and BLP are not used with LABEL, DATE, RETAIN; NS and BLP are invalid for output. If REEL is not specified for output files, standard labels are assumed. *REEL optional for output.
LABEL-filename (on tape) 'character string'	Opt	Opt	1-8 characters Not used with REEL-NS, -NL, or -BLP.
DATE-mmddyy ddmmyy	Opt	Opt	Not used with REEL-NS, -NL, or -BLP.
RETAIN-nnn	Opt	Opt	Code = 0.999; default = 0; not used with REEL-NS, -NL, or -BLP.
BLKL-block length	Opt	Opt	Block length 18-32767
RECL-record length	Opt	Opt	Record length 18-32767
RECFM-F FB V VB D DB	Opt	Opt	
END-LEAVE REWIND UNLOAD	Opt	Opt	Default is REWIND
DENSITY-1600 800 556 200	Opt 1600 800	Opt 800 556 200	Default is underlined
ASCII-YES NO	Opt	N/A	Default is NO; EBCDIC assumed if ASCII-NO
DEFER-YES NO	Opt	Opt	Default is NO.
CONVERT-OFF ON	N/A	Opt	CONVERT-ON is required if processing V, VB; Default is OFF.
TRANSLATE-OFF ON	N/A	Opt	Default is OFF.
PARITY-ODD EVEN	N/A	Opt	Default is ODD.
SEQNUM-X nnnn	Opt	Opt	Default is 1.

Combinations of 7-Track Specifications

Convert	Translate	Parity	
OFF	OFF	ODD	valid
OFF	OFF	EVEN	valid
OFF	ON	ODD	valid
OFF	ON	EVEN	valid
ON	OFF	ODD	valid
ON	OFF	EVEN	invalid
ON	ON	ODD	invalid
ON	ON	EVEN	invalid

FILE Statement (Multivolume Tape Files)

Function

The FILE statement supplies the system with information about files. The system uses this information to read records from and write records to tape.

Placement

You must supply a FILE statement for each new tape file that your program creates and for each existing tape file that your program uses. The FILE statement must follow the LOAD or CALL statement and precede the RUN statement.

Format

// FILE parameters

Contents

The FILE statement for processing multivolume tape files requires that you define and code the UNIT and REEL parameters differently than you would for single volume files. There are two reasons for this:

- When processing tape files contained on more than a single volume, the system requires information about each volume in order to perform all the necessary checking and protection functions.
- Additional information is needed to determine and check the sequence in which the volumes are processed and when they are to be mounted on the tape drives.

For multivolume tape files, the UNIT and REEL parameters of the FILE statement may require a list of codes. When you code a list of codes, the following rules apply:

- The list must be enclosed by apostrophes.
- The items in the list must be separated by commas.
- Intermixing 7- and 9-track units is not allowed.

The considerations for coding multivolume parameters are included in the following parameter discussions. The functions of the parameters are explained under *FILE Statement (Single Volume Tape Files).* Parameters not mentioned here are used as explained under *FILE Statement (Single Volume Tape Files).*

The maximum number of multivolume files allowed is explained under *Scheduler Work Area* in Part 2 of this manual.

REEL: The names of the tapes that contain or will contain the multivolume file must follow the keyword REEL. If the input tapes are not labeled or contain nonstandard labels, or if label processing is to be bypassed, the REEL parameter must be coded REEL-'NL,n', REEL-'NS,n', or REEL-'BLP,n', where n is the number of volumes in the file (99 volumes maximum). For output files, the n in REEL-'NL,n' is ignored.

UNIT: The keyword UNIT must be followed by a code or codes indicating the location of the tape unit that contains or will contain the file. No UNIT parameter may be repeated. The order of codes in the UNIT parameter must correspond to the order of names in the REEL parameter. When the number of codes in the UNIT parameter is less than the number of codes in the REEL parameter, the units are used alternately.

Examples

In the following examples, (A) shows a tape multivolume file consisting of three reels. The volumes must be mounted as follows:

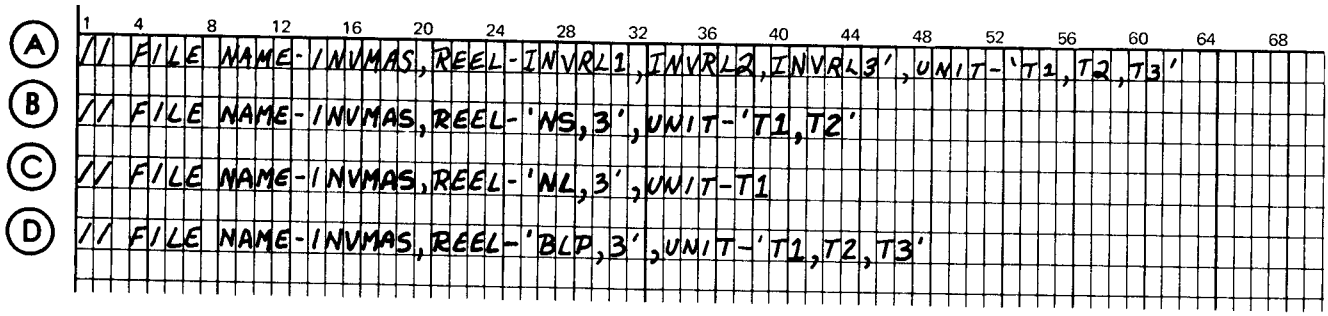
INVRL1 on tape unit T1
INVRL2 on tape unit T2
INVRL3 on tape unit T3

(B) shows a three-volume file with nonstandard tape label. The volumes must be mounted as follows:

First volume on tape unit T1
Second volume on tape unit T2
Third volume on tape unit T1

(C) shows a three-volume file with unlabeled reels. The volumes must be mounted in sequence on tape unit T1.

(D) shows the three-volume standard labeled file used in line (A), but with label processing bypassed.



Spooling Considerations

None

FILE Statement (Multifile Tape Volumes)

Function

The FILE statement supplies the system with information about tape files. The system uses this information to read records from and write records to tape.

Placement

You must supply a FILE statement for each new tape file that your program creates and for each existing tape file that your program uses. The FILE statement must follow the LOAD or CALL statement and precede the RUN statement.

Format

// FILE parameters

Contents

More than one file can exist on a reel of tape. If this is the case, each file has an associated file number. The SEQNUM parameter indicates the file number and must be specified on the tape FILE statement. The system positions the tape to the desired file using this parameter. Only the parameters that pertain to multifile tape volumes are discussed. All other tape FILE parameters are described in *FILE Statement (Single Volume Tape Files).*

The SEQNUM keyword for multifile volume tape files has the following parameters:

SEQNUM-nnnn
-X

where nnnn = file sequence number, assigned when the file was written. Default value is 1. The number indicates the relative position of the file in a volume of files and is incremented by one from one file to the next.

X = prepositioned file. A user routine or END-LEAVE has prepositioned the tape to the desired place for input or output.

END-LEAVE, when coded on the FILE statement, causes tape to be left (not rewound) at the place where processing was completed.

If several files are to be processed from a single volume, the user can decrease file allocation time by (1) coding END-LEAVE on the FILE statement for each of the files that are to be accessed, and (2) processing the files in the order in which they reside on tape.

Prepositioned Tapes (SEQNUM-X on the FILE Statement)

The user can preposition a file for input or output and, in a subsequent job or job step, begin file processing at the prepositioned point. The prepositioning can be accomplished several ways. For example:

- An RPG II program can process an input tape and, at some point, set on the LR indicator to force end of job.
- The same technique can be used by object programs produced by other compilers (or the assembler), that is, force (or call) end of job before the input file has been processed to end of file.

END-LEAVE would have to be coded on the tape FILE statement, or else the tape would be rewound at end of job.

When tape has been prepositioned, for input or output, SEQNUM-X must be coded on the tape FILE statement.

If a standard labeled tape has been prepositioned, label verification will not be performed by the system. That is, the file label, date, and volume sequence information in the HDR1 label will not be checked nor will the record format, block length, record length, and recording technique (translate, convert, parity) information in the HDR2 label be checked. These fields are normally compared to the FILE statement and/or program DTF. Label verification will be performed, however, if the tape was prepositioned to the start-of-a-file (before the HDR1) record.

Restrictions on the Use of Multifile Tapes

The following restrictions must be adhered to when multifile tape volumes are used:

- All files in the volume must be labeled in the same manner, that is, all standard labeled files or all unlabeled files.
- All files in the volume must be recorded in the same density.
- All files must be recorded in the same mode (translate, convert, and parity).
- If the last file on a multifile reel is continued on a subsequent reel, the two reels constitute an aggregate. The preceding three restrictions apply to all volumes of the aggregate. In addition, all volumes of the aggregate must be either 7- or 9-track.
- Standard labeled 7-track tapes, if prepositioned, should be prepositioned to a point just before a HDR1 record. Otherwise tape data checks or runaway may occur.

Note: If standard labeled tapes (7-track, primarily) have been prepositioned to any point other than the label, you can access the data, without the above errors, by coding SEQNUM-X, REEL-NL on the FILE statement. You should *not,* however, output a file on a standard labeled tape in this manner because the file would not be accessible by the system (there cannot be a mixture of labeled and unlabeled files on the volume).

The following sections describe the use of standard labeled, nonnstandard labeled, and unlabeled multifile tape volumes.

Standard Labeled Files

Standard labeled multifile tape volumes have the following format:

VLG HLG TM DATA TM TLG TM
HLG TM DATA TM TLG TM TM

where

VLG = volume label group VOL1 label is processed; others, if present, are ignored.

HLG = header label group HDR1, HDR2 records are processed; other header records, if present, are ignored.

TM = a single tape mark

DATA = the data file

TLG = trailer label group EOF1, EOF2 records, when processed, indicate the end of a file. EOV1, EOV2 records indicate the end of volume of a multi-volume file. The final volume of the file will have EOF1, EOF2 trailer records; other trailer labels, if present, are ignored.

Two tape marks indicate the end of volume if the preceding TLG consisted of EOF1, EOF2 records.

One tape mark indicates end of volume if the preceding TLG consisted of EOV1, EOV2 records.

Each file is preceded by a header label group and a tape mark, and is followed by a trailer label group and a tape mark.

The HDR1 record contains a field in which the file sequence number is recorded when the file is written. The number indicates the relative position of the file in the volume, and it increments by one from one file to the next.

The user selects the desired input file by specifying on the FILE statement:

SEQNUM-number

where number = the file sequence number in the HDR1 record of the desired file. Default value is 1.

For output files, SEQNUM-number specifies where the file is to be written on the volume. The number specified should be one greater than the last file on the volume, unless you want to overlay a file. A system message occurs if the SEQNUM value is two or more greater than the value in the HDR1 record of the last file on the volume. Once a file has been written, any files that existed on the volume beyond the new output file are no longer accessible for input. Two tape marks are written after the new file, indicating the end of volume.

For output files, the tape is positioned per the SEQNUM parameter, then the trailer label of the preceding file is read. If the label group is EOV (end of volume) instead of EOF (end of file), a diagnostic halt message is issued; a file cannot be written after a multi-volume file.

If a file exists where the new output file is to be written, the HDR1 label of the file is read. The expiration data is checked; if the file has not expired, a diagnostic halt is issued.

If a series of files is being built on one output tape in one job stream, you can code SEQNUM-1, END-LEAVE on the FILE statement for the first file. Code SEQNUM-X and END-LEAVE on the FILE statement for the second and subsequent files. When the job stream is run, the system assigns the SEQNUM value for files built after the first file. The value assigned will be one greater than that used for the previous file.

Notes:

1. If files exist on the tape *beyond* the one being written, they cannot be accessed after the file is written.
2. If unexpired (or permanent) files exist beyond the file being written, they are not detected and are lost.
3. For input files, a system message occurs if a file having the specified SEQNUM value cannot be found.
4. For output files, if the tape has been positioned within the boundaries of an existing file, a diagnostic message occurs.

If the tape has been prepositioned for input, SEQNUM-X should be coded on the FILE statement. The system then determines where the tape is positioned as follows:

1. If at load point, the volume and file header labels are read and verified. The tape then is positioned to the start of the data.

2. If at the start of a file, the file header labels are read and verified. The tape is positioned to the start of the data. The volume label is not read.

3. If the tape was prepositioned to any other point, the volume or file header labels are not checked. The tape is not positioned to any other point; it is assumed that the data area starts where the tape is currently positioned.

Note: File allocation time can be decreased when several files on one volume are to be processed by a job by (1) coding END-LEAVE on the FILE statement for each file referenced on the volume and (2) by processing the files in the order in which they reside on tape.

Nonstandard Labeled Files

The REEL parameter on the tape FILE statement indicates a nonstandard labeled file when REEL-NS is used. Nonstandard label groups cannot be written but are accepted in input files. A nonstandard label group is one in which the first record is not an 80-byte record beginning with VOL1. The label group is followed by a tape mark.

For single-file volumes, the system positions the tape past the tape mark that follows the label group. This is done after the first record on the reel has been verified as not being an 80-byte VOL1 record. The label group is not read or processed. Trailer labels, if present, are not read or checked when the file is closed.

Multifile nonstandard labeled volumes must be prepositioned before the job is run if the file to be processed is not the first file on the volume. If the tape file has been pre-positioned, it is indicated by coding SEQNUM-X on the tape FILE statement. The system does not check the tape; the file is allocated. It is the user's responsibility to position the tape properly.

Unlabeled Volumes

Unlabeled multifile tape volumes have the following format:

DATA TM DATA TM DATA TM TM

where TM = a single tape mark denoting the end of a file.

Two tape marks denote the end of a volume unless the last file is a multivolume file, in which case there is only one tape mark.

A single tape mark may precede the first file of an input volume. The first file on an output volume does not have a leading tape mark.

The tape can be positioned to any file on the reel, for input or output, by use of SEQNUM-number on the tape FILE statement. The number indicates the relative position of the file on the tape. If SEQNUM-X is used for an output file, the system ensures that the tape is positioned at the start of a file before writing. (The tape is backspaced to the start of the file if necessary.) When SEQNUM-X is used for an unlabeled input file, the system does not move the tape before it is read.

For output files the SEQNUM value should be one greater than the number of files on the reel, unless you wish to overlay a file.

For input or output files, specifying a SEQNUM value two or more greater than the number of files on the reel will cause either (1) a message to be issued if the last file on the tape is not a multivolume file or (2) unpredictable results if the last file on the tape is a multivolume file.

Allocation time can be reduced when several files on a reel are to be accessed during a job by (1) processing the files in the order in which they reside on the tape and (2) coding END-LEAVE on the FILE statement for each file. Then the tape is not rewound for each allocation. However, if SEQNUM-X is coded on one FILE statement and a subsequent FILE statement has SEQNUM-number coded for the same tape volume, the volume is rewound by the system before being positioned to the desired file.

Note: Standard labeled tapes can inadvertently be destroyed when SEQNUM-X and REEL-NL are used for an output file. If the tape is not at load point, a check for a standard labeled tape is not performed.

REEL Parameter on FILE Statement

The REEL parameter, for standard labeled tapes, is the name of the volume containing the file.

REEL-name

If a multifile multivolume aggregate is used, REEL-name is the name of the *first* volume of the aggregate, regardless of which volume actually contains the file, when a single-volume file is accessed.

The parameters used for multivolume files are:

REEL-'name1,name2,namex'

where name1 is the name of the first volume of the aggregate (as described above), and name2 and namex are the volume names of the additional volumes containing the file.

The names to use in this case are the actual volume names of each volume.

Spooling Considerations

None

FILE Statement (Device Independent Files)

Function

The device independent FILE statement allows you to assign I/O devices used by a program. This statement supplies the system with information about I/O devices used in the program. This information is used to read records from and/or write records to the I/O devices used.

Placement

You must supply a device independent FILE statement for each device independent file used in your program. The device independent FILE statement must follow the LOAD or CALL statement and precede the RUN statement.

Format

// FILE parameters

Note: If device independent files are using disk or tape units for input or output, then the disk or tape FILE statement format must be used.

Contents

All parameters are keyword parameters. The parameters are as follows (keywords are in capital letters):

NAME-filename (in program)

Unit-code

PRINT-code

RECL-record length

The NAME and UNIT parameters are always required on the device independent FILE statement. This FILE statement format is used when device independent files are being read from or written to unit record devices.

Programs requiring specific filenames for device independent files are:

Program	File	Name
Disk Sort	Input	INPUT or INPUT1 INPUT2 through INPUT8
Copy/Dump	Input Output	COPYIN COPYO COPYP
System History Area Display	Output	$HISTORY
Spool File Copy	Input Input Output Output Output Output	READERQ[1] (optional) RESTORE[1,2] (optional) DISPLAYQ[1] (optional) PRINTQ[1] (optional) PUNCHQ[1] (optional) READERQ[1] (optional)

[1]The file name can be replaced by the name specified on a control statement parameter.
[2]The RESTORE file must be either tape or disk.

NAME: The NAME parameter is required on the device independent FILE statement. It tells the system the name the program uses to refer to the file. The name can be any combination of characters except commas, apostrophes, or blanks. The first character must be alphabetic, and the name used may not exceed 8 characters in length.

UNIT: The UNIT parameter is required on the device independent FILE statement. It tells the system the unit record device from which the file will be read or to which the file will be written. The UNIT parameter must be one of the following:

MFCU1	Primary hopper of the 5424 (input and output files)
MFCU2	Secondary hopper of the 5424 (input and output files)
MFCM1	Primary hopper of the 2560 (input and output files)
MFCM2	Secondary hopper of the 2560 (input and output files)
1442	1442 Card Read Punch (input and output files)
2501	2501 Card Reader (input files only)
1403	1403 Printer (output files only)
3284	3284 Printer (output files only)
3741	3741 Data Station/Programmable Work Station (input and output files)
READER	Use the partition's assigned system input device (input files only).
	Note: You cannot use this parameter if the system input device is assigned to the console.
PRINTER	Use the partition's assigned system print device (output files only)
PUNCH	Use the partition's assigned system punch device (output files only)
	Note: This parameter cannot be used if the system has no punch device.

PRINT: The PRINT parameter is used to specify whether interpreting is to be done on punch files. It is ignored when the file is not being punched or when the punch device does not support printing. If PRINT-YES is specified, the data being punched is also printed on the card. When PRINT-NO is specified, the data is not printed. If this parameter is not specified and the device is capable of printing, PRINT-YES is assumed.

RECL: The RECL parameter is used to specify the number of bytes in a logical record on the 3741. It can be any number from 1 through 128. If a device other than the 3741 is used, the parameter is ignored. If this parameter is not specified for the 3741, a record length of 96 is assumed (unless overriden by the program).

Spooling Considerations

None

HALT Statement

Function	The HALT statement is used to override the system nohalt mode established at IPL or the nohalt mode established by a NOHALT statement (see NOHALT statement or or NOHALT command). The system normally does not stop at the end of job or job step in a partition. The HALT statement causes the partition in which it was received to stop whenever a system message is issued or when end of job or job step occurs, and requires the operator to select the recovery option. The operator must restart the partition by responding to the system message issued. The system remains in halt mode until a NOHALT statement is issued, IPL occurs, or end of job is reached. It then returns to the IPL nohalt mode or the mode established by the last HALT/NOHALT command received by the system. When a HALT OCL statement is received in a partition, the severity code for halts is reset so all system halts including end-of-step and end-of-job halts must be responded to. See *NOHALT Statement* for a description of severity codes.
Placement	A HALT statement can be placed anywhere among the OCL statements. In a procedure it must precede the RUN statement.
Format	// HALT
Contents	None (comments may be entered after the blank position following HALT).
Spooling Considerations	None

IMAGE Statement

Function

To operate correctly, the printer requires characters matching those on the printer chain or train to be in a special area of main storage called the chain-image area. When you replace the printer chain or train with one having different characters, you must also change the contents of the chain-image area.

The IMAGE statement instructs the system to replace the contents of the chain image area with the characters indicated by the data records entered from the system input device or contained in a source library. The effect of the IMAGE statement is temporary, and the system chain image is returned to the chain image specified during system generation if the system IPL procedures is performed again.

Placement

The IMAGE statement can appear anywhere among the OCL statements. In a procedure, it must precede the RUN statement.

Format

// IMAGE parameters

Contents

The IMAGE statement tells the system one of two things: (1) the new chain characters are to be read from the system input device, or (2) the new chain characters are to be read from the source library.

The IMAGE parameters are:

format-{HEX / CHAR / MEM}

number-value

name-name

unit-code

Note: The words *format, number, name,* and *unit* are not coded as part of the parameters for the IMAGE statement.

(Coding only HEX, CHAR, or MEM is preferable for format, but HEXADECIMAL, CHARACTER, or MEMORY can be coded.)

Characters from the System Input Device

If you wish to indicate that new characters are to be read from the system input device, use the following parameters:

format: Use the word CHAR to indicate that characters are in EBCDIC form. Use the word HEX to indicate that the characters are in hexadecimal form.

number: The number parameter must be used with HEX and CHAR. It must be a value that is equal to the number of columns or line positions in the data records or the data keyed in following the IMAGE statement that contains the new characters. This number must not exceed 240 when the characters are hexadecimal or 120 when characters are EBCDIC. The name and unit parameters must not be coded. The rules for punching and keying the new characters are as follows:

- The characters must begin in position 1.
- Consecutive card columns or line positions must be used; however, only the first 80 columns or line positions of the card or line can be used. Hexadecimal requires an even number of columns or line positions, two per character.
- Characters continued on another card or line must begin in position 1.

Characters from the Source Library

To indicate that new chain characters are to be read from the source library, the MEM parameter must be specified. The following parameters must also be included:

Name: The name parameter identifies the source member containing the characters in the library. You can place the characters in a source library by using the library maintenance program. The name you supply in the library maintenance control statement is the name used to identify the characters in the source library.

Unit: The unit parameter must be used with the name parameter. It tells the system which simulation area contains the source library. Possible codes are R1, F1, R2, F2.

Examples

The IMAGE statement in (A) tells the system that the new characters (from the system input device) are in hexadecimal form; the number parameter indicates that there are 120 positions (60 characters) containing the new characters.

In (B), the new characters (from the system input device) are in EBCDIC. The number parameter indicates that there are 48 columns or line positions containing the new 48-character image.

(C) tells the system that the new characters are in the source library. The name parameter indicates that the characters were named CHAIN in the source library. The unit parameter indicates that the source library containing them is on the simulation area assigned to R1. Examples of the entry specified in (C) are (A) and (B). The entry itself must contain an IMAGE statement with the characters in either hexadecimal or EBCDIC. The number of columns containing the characters must also be specified. (See *Library Maintenance Program* in Part 4 for restrictions on the name used in coding MEM.)

(A)

```
1   4   8   12  16  20  24  28  32  36  40  44  48  52  56  60
// IMAGE HEX,120
F1F2F3F4F5F6F7F8F9F0E7E861E2E3E4E5E64F7A6D7F6B7ED1D2D3D4D5D6
D7D8D960E94DC1C2C3C4C5C6C7C8C94E4B5D6C5B5C7B507C4C5E5F7D6F6E
```

(B)

```
1   4   8   12  16  20  24  28  32  36  40  44  48  52  56  60
// IMAGE CHAR,48
1234567890#@/STUVWXYZ&,%JKLMNOPQR-$*ABCDEFGHI+.'
```

(C)

```
1   4   8   12  16  20  24  28  32  36  40  44  48  52  56  60
// IMAGE MEM,CHAIN,R1
```

Spooling Considerations

No support is provided by spooling for changing the chain image for printed output. Print records that are spooled are printed with the currently loaded chain image. If a special print chain or train is required for a job step on the print queue, it is the operator's responsibility to load the chain image before the print writer starts printing the output.

If you want to change your chain image in a spooling environment, use one of the following methods:

- Supply an IMAGE statement in a partition. This changes the chain image for the print writer but not for the job being loaded in the partition. The print writer must be stopped with a STOP command before the chain image is changed.
- Bypass spooling, allowing the partition to use the printer directly by using the START SPOOL and STOP SPOOL command in the partition and START PRT and STOP PRT command.

INCLUDE Statement

Function

The INCLUDE statement identifies the entry (procedure) in the source library that contains the OCL statements to be merged into the job stream. When printed, the merged OCL statements are identified by an X/ preceding the statement identifier. The source member to be included into the job stream cannot contain a CALL statement.

Placement

The INCLUDE statement can appear anywhere among the OCL statements. It must precede the RUN statement (if RUN is used) in a procedure. Nested INCLUDE statements are not allowed.

Format

// INCLUDE procedure-name,unit[,switch characters]

Contents

Procedure-name: This name identifies the procedure in the source library. It must be the same name that was supplied in the library maintenance control statement when the procedure was entered in the source library.

Unit: The unit parameter is a required code. The code identifies the simulation area that contains the procedure. Possible codes are R1, F1, R2, F2.

Switch characters: The switch characters (0, 1, and X) are optional. When you include them, you must supply 8 characters, because they are compared with the eight external indicators. The system does a comparison for each position if the switch character is a 0 or 1. An X cancels the compare operation for that position only. The first (leftmost) switch character is compared with external indicator 1; then the second switch character is compared with external indicator 2; this process continues until the 8 switch characters and the eight external indicator positions are either compared or bypassed. If an equal condition exists, the INCLUDE statement is accepted. Otherwise, an informational message is displayed and the INCLUDE statement is not accepted.

Example

```
// SWITCH 10101010
// LOAD ABC,F1
1 // INCLUDE X,R1,10101010
2 // INCLUDE Y,R1,01010011
3 // INCLUDE Z,R1,X0X0X010
// RUN
```

1 This INCLUDE statement is accepted, and the OCL statements in procedure X are merged into the job stream.

2 This INCLUDE statement is not accepted, and an informational message is displayed.

3 This INCLUDE statement is accepted, and the OCL statements in procedure Z are merged between procedure X OCL statements and the RUN statement. The job stream is changed as follows:

```
//   SWITCH 10101010
//   LOAD ABC,F1
//   INCLUDE X,F1,10101010
X/   DATE 051077                                  }
X/   FILE NAME-A,UNIT-D1,PACK-D1D1D1  } Procedure X
X/   FILE NAME-B,UNIT-D1,PACK-D1D1D1  }
//   INCLUDE Y,F1,01010011
     CR FL SH I                    ABC01
**   THIS INCLUDE BYPASSED ** INDICATORS-10101010**
//   INCLUDE Z,F1,X0X0X010
X/   FILE NAME-D,UNIT-D2,PACK-D2D2D2  }
X/   FILE NAME-E,UNIT-D2,PACK-D2D2D2  } Procedure Z
X/   FILE NAME-F,UNIT-D2,PACK-D2D2D2  }
//   RUN
```

Spooling Considerations

None

JOB Statement

Function

The JOB statement allows you to group related job steps together to ensure that they are run sequentially. The system does not initiate the next step of a job until the previous job step has been completed successfully. It is required on systems that have active spooling or on nonspooling systems running in job mode.

Placement

The JOB statement must precede the first LOAD or CALL statement. It cannot be used in a procedure.

Format

//jobname JOB parameters

Jobname: This is a required entry used to uniquely identify a job. It must begin in position 3, may not exceed 8 characters in length, and may not contain a comma. When the last two characters of a jobname are asterisks (**), the system replaces them with a sequence number. The assigned number (from 01 through 99) is included in an informational message.

Job (and step) names should contain only characters that are on the 3277 Display Station keyboard. If the jobname contains characters that are not on the 3277 keyboard, that job cannot be referenced by its jobname by using commands. These nondisplayable characters may also cause display problems when written to the 3277 display. The jobname is displayed on the display screen with system messages and halts associated with the job. It is also the name by which jobs are identified on the spooling queues.

Content

All parameters are optional. They are as follows (defaults are underlined):

PRIORITY-{0, 1, 2, 3, 4, 5}

CORE-nnn (can be 1, 2, or 3 digits)

SPOOL-{YES, NO}

PARTITION-{1, 2, 3, A, B, C, D}

QCOPY-{YES, NO}

PRIORITY: A priority may be assigned to a job to indicate its level of importance on the reader or output queues. The priority of the job steps for a job on the output queues is the priority assigned to that job on the reader queue unless overridden by a PRIORITY parameter on a PRINTER or PUNCH statement. Priority 0 indicates a priority 1 job, which is to be held on the reader queue until released via an operator control command. (See Appendix C for a summary of operator control commands.) Priority 5 is the highest priority that may be assigned. Within a given priority, jobs are scheduled on a first-in, first-out basis. If this parameter is not specified, priority 1 is assumed.

CORE: This parameter specifies the amount of main storage required to execute the largest step in a job and establishes a minimum partition size for the duration of that job. (It does not assign main storage to the partition; a SET command must be used for this purpose.) When this parameter is not specified, the system assumes that the job is scheduled in a partition with adequate main storage and uses the present partition size as the minimum for the job. Eight is the minimum value that may be specified; 238 is the maximum value that may be specified. Anything less than 8K is treated as 8K. This can be increased in 2K increments up to a maximum of 238K. If the parameter is not a 2K increment, it is rounded up to the next 2K increment. When the CORE parameter is specified, the partition must have the requested amount of main storage available before the job can be executed.

Note: If more than 128 files are to be processed in a partition, minimum partition size is 10K.

SPOOL: SPOOL-NO ensures that the job will not be loaded if spooling is active in that partition. If SPOOL-NO is specified, the system prevents the job from being executed when spooling is active. It is your responsibility to ensure that I/O devices for the job are available and spooling has been stopped for the partition. You may free a device required by a job and stop spooling in a partition by using a STOP command. (See Appendix C for a summary of operator control commands.) SPOOL-NO should be used when a job requires dedicated use of one or more devices being used by spooling. SPOOL-YES specifies a job that may be run with spooling active. If this parameter is not given, SPOOL-YES is assumed.

PARTITION: This parameter is used to specify the partition in which the job must be executed. If this parameter is not specified and input spooling is not used, the system assumes the job can be executed in any partition. If PARTITION is not specified and input spooling is being used, the job is scheduled for execution in partition 1. Once a job is placed on the reader queue, its partition assignment can be changed through use of the CHANGE partition commands.

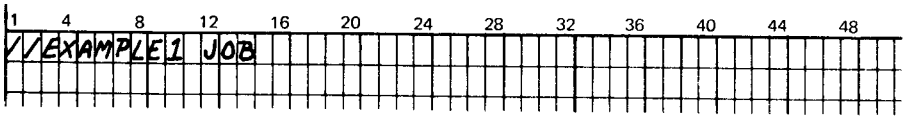
The meaning of each partition code is:

Code	Meaning
1	Execute job in partition 1
2	Execute job in partition 2
3	Execute job in partition 3
A	Execute job in partition 1 or 2
B	Execute job in partition 1 or 3
C	Execute job in partition 2 or 3
D	Execute job in partition 1, 2, or 3

Note: When keyword parameters are not specified on this statement, comments may not be given following the JOB statement identifier.

QCOPY: This optional parameter (QCOPY-NO) prevents the spool file copy program ($QCOPY) from accessing the job on the reader queue. If QCOPY-YES either is specified or is the default, the spool file copy program can copy the job from the reader queue.

This page intentionally left blank.



The preceding JOB statement causes the following:

- Statement read by the spooled reader:
  - Priority 1 is assigned to the job on the reader queue and output queues (see note).
  - Job is executed in partition 1.
  - EXAMPLE1 is the jobname on the reader and output queues.

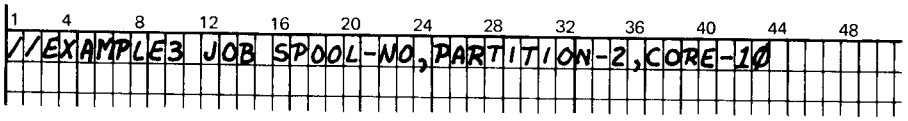
- Statement is not read by the spooled reader:
  - The partition the job executes in is the partition into which the operator loads the job.
  - If spooling is active for the partition into which the job is loaded, priority 1 is assigned to the output queues (see note).

The preceding JOB statement causes the following:

- Statement read by the spooled reader:
  - Priority 5 is assigned to the job on the reader queue and output queues (see note).
  - Job is executed in partition 1.
  - EXAMPLE2 is the jobname on the reader and output queues.
  - Prior to the job being initiated, 48K of main storage must be available in partition 1.

- Statement not read by spooled reader:
  - The partition the job executes in is the partition into which the operator loads the job.
  - The job is executed if 48K of main storage is available in the partition into which the job was loaded.
  - If spooling is active for the partition into which the job is loaded, priority 5 is assigned to the output queues (see note).

Note: The priority on the output queues can be different for a job step within the job if the PRIORITY parameter is specified on the PRINTER and/or PUNCH statement.



The preceding JOB statement causes the following:

- System with spooling
  - The job must be loaded into partition 2.
  - Spooling may not be active in partition 2.
  - At least 10K of main storage must be available in partition 2.

- System without spooling
  - The job must be loaded into partition 2.
  - At least 10K of main storage must be available in partition 2.

Spooling Considerations

If spooling is active in a partition, the JOB statement must be used in that partition. When jobs are being read onto the reader queue, any errors on the JOB OCL statement cause that job to be assigned a priority of 5. If the jobname is missing or invalid, the default jobname, JOB, is assigned to that job on the reader queue.

LOAD and LOAD * Statement

Function

The LOAD statement identifies the program to be executed and indicates whether the program will be loaded from the system input device for the partition, from an object library, or from a file on a main data area.

Placement

One LOAD statement is required for each program executed. On systems being run in step mode, the only requirement is that the LOAD statement precede the RUN statement. The LOAD statement must follow the JOB statement, and precede the RUN statement when operating in job mode. Any number of job steps (LOAD/RUN combinations) may follow a JOB statement. For a complete description of job mode and step mode see Part 2 of this manual.

Format

The LOAD statement has the following formats:

```
//[stepname] LOAD * [,program-name1,unit1]
[,switch characters] (a blank is mandatory between LOAD and *.)

//[stepname] LOAD program-name2,unit2
[,switch characters]
```

The first format is used to load object programs from the system input device or from a file on a main data area. The second format is used to load object programs from an object library.

Contents

stepname: This optional entry is used to uniquely identify a job step. If specified, the stepname must begin in position 3 of the statement, must not exceed 8 characters in length, and must not contain a comma. Stepnames (and jobnames) should contain only characters that are on the 3277 Display Station keyboard. If the stepname contains characters not on the 3277 keyboard, that step cannot be referenced by its stepname by using commands. These nondisplayable characters may also cause display problems when written to the 3277 display. The stepname is displayed on the display screen with system messages associated with the job step. If a stepname is not specified, the system assigns a stepname to each step. The stepname assigned by the system is made up of the program name from the LOAD statement and a 2-digit number assigned by the system. Stepnames assigned by the system are incremented by one at end of job step in which a stepname is assigned. If a LOAD * statement without a stepname is encountered, the system assigns a stepname of ASTRSKnn. The number portion of the stepname is reset to 1 at end of job. After 99 stepnames have been assigned within one job, the number is reset to 1. When the print and punch queues are displayed, the stepname identifies job steps on the queues.

Asterisk: An asterisk is specified when the user wants the object program loaded from the partition's system input device or from a file on a main data area. The object program must follow the RUN statement if the program is loaded from the partition's system input device, and a /* statement must follow the object program to indicate the end of the object program input. The object program is temporarily copied into the object library on the system pack and then loaded into main storage for execution. LOAD * programs are accepted in any partition; however, while a LOAD * program with overlays is running, no other LOAD * programs can be loaded. When a subsequent LOAD * program is received, the system issues a diagnostic, which allows the user either to wait for completion of the LOAD * program or to cancel the subsequent LOAD * program.

Program-name1: This entry identifies the file that contains the object program to be loaded. The program-name1 can be any combination of characters except commas, apostrophes, or embedded blanks. The first character must be alphabetic. The length must not exceed 6 characters.

unit1: This entry specifies the main data area that contains the file from which the object program is to be loaded. Possible codes are those for the main data areas.

Note: Although the user does not specify an OCL FILE statement, the LOAD * from file function requires an F1 entry in the SWA. The system produces a simulated FILE statement to protect the file from which the program is being loaded. Therefore, the number of available file statements that can be used when a program is loaded from a file is reduced by 1. The program-name1 parameter must identify the file which contains the program being loaded.

Switch characters: The switch characters (0, 1, and X) are optional. When you include them, you must supply 8 characters because they are compared with the eight external indicators. The system compares each position if the switch character is a 0 or 1. An X cancels the compare operation for that position only. The first (leftmost) switch character is compared with external indicator 1; then the second switch character is compared with external indicator 2. This process continues until the 8 switch characters and the eight external indicator positions are either compared or bypassed. If an equal condition exists, the program is loaded. Otherwise, an informational message is displayed and the job stream is flushed to the next step.

The program-name1 and the unit1 parameters are positional parameters. Therefore, if you include the switch characters without these parameters, you must separate the * and the switch characters with either 1 or 3 commas. For example:

```
// LOAD *,xxxxxxxx
  or
// LOAD *,,,xxxxxxxx
```

program-name2: The program-name2 is the name used to identify the program in the object library. Program-name2 may be up to 6 characters in length. The name must begin with one of the 29 alphabetic characters (A–Z, @, #, or $) and may be followed by up to 5 characters. Commas, apostrophes, periods, and blanks may not be used in the name. The system service programs and program products are identified by the following names:

Program	**Name**
Restart	$$RSTR
Alternate Track Assignment	$ALT
Basic Assembler	$ASSEM
RPG II Auto-Report	$AUTO
Alternate Track Rebuild	$BUILD
COBOL Compiler	$CBL00
Communication Control Program	$CCP
Card List	$CLIST
Configuration Record	$CNFIG
Copy/Dump	$COPY
Card Sort/Collate	$CSORT
Dump/Restore	$DCOPY
File Delete	$DELET
CCP/Disk Sort	$DGSRT

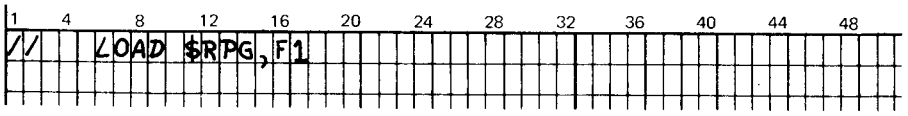
Program	Name
Disk Sort	$DSORT
File Compress	$FCOMP
FORTRAN	$FORT
Gangpunch	$GANGP
System History Area Display	$HIST
Disk Initialization	$INIT
Chain Cleaning	$KLEAN
File and Volume Label Display	$LABEL
Library Maintenance	$MAINT
Macro Processor	$MPXDV
Overlay Linkage Editor	$OLINK
Spool File Copy	$QCOPY
Card Reproduce and Interpret	$REPRO
Recover Index	$RINDX
RPG II Compiler	$RPG
Reassign Alternate Track	$RSALT
Simulation Area	$SCOPY
Tape Initialization	$TINIT
Tape Sort	$TSORT
Tape Error Summary Program	$TVES
VTOC Service	$WVTOC

Unit2: The unit2 parameter is a required code. It indicates the simulation area that contains the program. Possible codes are R1, F1, R2, F2.

The disk area containing your object program is called an object library. You can create an object library on any of the simulation areas by using the library maintenance program. (See *Library Maintenance Program* in Part 4 of this manual.)

Example

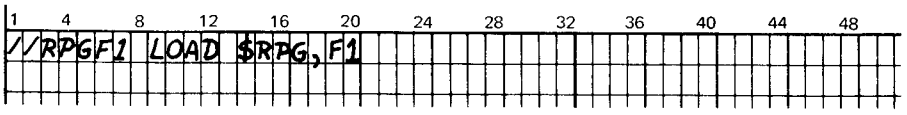
In the following sample LOAD statement, $RPG is the name that identifies the RPG II compiler.



F1 is the code indicating the simulation area where the compiler is located.

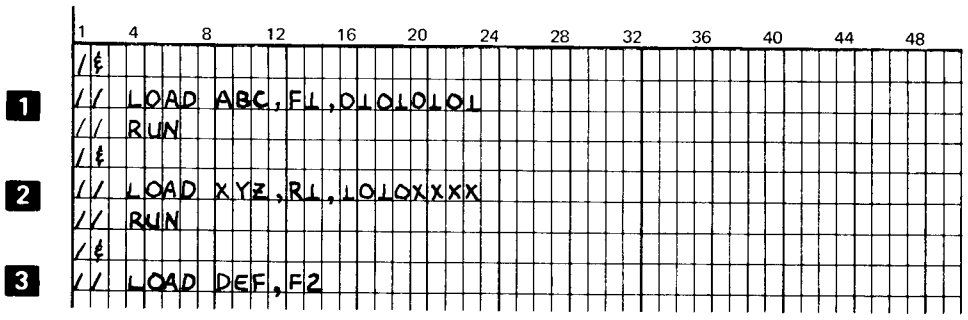
The system would assign a stepname of $RPG01, because a stepname is not specified on the LOAD statement.

The following example shows a stepname assigned by the user:



The system would use the stepname RPGF1.

The following example shows how the switch characters can be used to determine whether or not a job step will be executed. Assume that a program has just completed executing successfully and has set the external indicators to 10101010. The OCL statements following the completed program are:

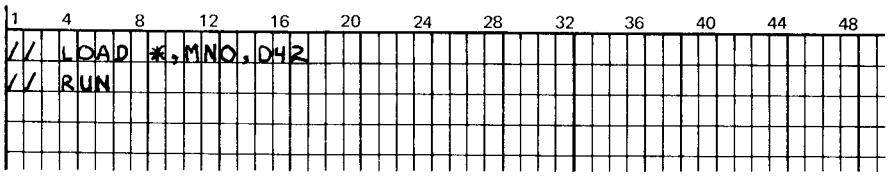


1 Step 1 is not executed because the external indicators do not agree with the switch characters. An informational message with the value of the external indicators is issued. The system then flushes to the next step (step 2).

2 Step 2 is executed because the external indicators agree with the switch characters. (The switch character X tells the system to cancel the compare operation with the relative external indicator.)

3 Step 3 is executed after step 2 is completed.

The following example shows the OCL statements and parameters needed to load a program from a file:



The LOAD * identifies the requested function. The program is processed as a LOAD * type program. The same rules and restrictions that apply to a program loaded from the system input device also apply to a program that is loaded from a file.

The program-name (MNO) must be the name of the file that contains the program.

The program is contained in a file that is located on a main data area (D42). The program is copied to the object library on the system pack before being loaded into main storage for execution. The system pack becomes the program pack for all LOAD * programs regardless of whether they are loaded from disk or from the system input device.

Spooling Considerations

None

LOG Statement

Function

The LOG statement is used to perform two functions:

- Change the system log device for a partition.
- Control end-of-job page ejection on the log device.

System Log Device

During system generation, a system log device is established for each partition. When a LOG statement is used, the device specified as the system log device remains in effect for the partition until another LOG statement is read or until IPL is performed. The log device applies only to the partition in which the LOG statement is processed.

The following information is *not* logged on the printer, but it *is* logged in the system history area (SHA):

- Responses to ERP (error recovery procedures) messages
- Responses to EJ/ES messages
- Spooling messages and responses
- Volume recognition facility messages
- OCC
- OCC diagnostics
- Any ERP message received while a printer ERP message is active
- Partition identification on OCL statements and control statements for system service programs.

If logging of the above information is required, you should assign the log device only to the CRT and periodically print the contents of the system history area using the system history area display program ($HIST). See *System Service Programs* in Part 4 for a description of $HIST.

Logging to the SHA

All system messages and OCL statements are logged to an area on the system pack called the SHA (system history area).

Note: You may print the SHA or copy the SHA to a device independent file using $HIST. Also, you may display the SHA on the display screen using the DISPLAY command. See *System Service Programs* in Part 4 for a description of $HIST, or Appendix C for a summary of the operator control commands.

Logging to the CRT

System messages that require a response and informational messages are logged to the CRT independent of the assigned log device. The only effect of a // LOG CONSOLE statement is that logging to the printer is stopped.

Logging to the 3284

If the 3284 is assigned as the log device and is not assigned to a partition, system messages, OCL, and control statements are logged to the 3284.

Logging to the 1403

System messages, OCL, and control statements are logged to the 1403 if the 1403 is assigned as the system log device, if the 1403 is *not* assigned to a partition, and if one of the following conditions is met:

- Print spooling is neither active nor using the 1403. In this case, OCL statements and messages are logged directly to the 1403.
- Print spooling is active (intercepting print requests). In this case, OCL statements and system messages are logged to the printer queue.

Controlling Page Ejection

The LOG statement is used to control page ejection that occurs before ES and EJ and after EJ. The EJECT and NOEJECT modes remain in effect for any log device until another LOG statement is read, or until an IPL is performed.

Placement

You can use the LOG statement within any of the sets of OCL statements for your job steps. In a procedure, it must precede the RUN statement.

Format

```
// LOG device[,mode]
   or
// LOG ,mode
```

Contents

The following codes can be used as parameters.

Device	Meaning
CONSOLE	Log to the CRT and the system history area.
1403	Log to the CRT, the 1403 printer, and the system history area.
3284	Log to the CRT, the 3284 printer, and the system history area.
none	The log device is not changed.

Mode	Meaning
EJECT	Eject a page before ES and EJ and after EJ. If no mode is specified, EJECT is assumed.
NOEJECT	Do not eject a page before ES and EJ and after EJ.

Spooling Considerations

When 1403 printed output is being spooled, and the system log device has been assigned to the 1403 printer, all system messages that would normally be logged on the 1403 are placed in the spooled print file. When the spooled output is printed, the system messages are printed, along with the job steps' output. When spooling 1403 output, a page eject occurs at the start of every job step's printed output, regardless of the mode specified on the LOG statement.

NOHALT Statement

Function

The NOHALT OCL statement is used to cancel the halt mode established by a HALT OCL statement. It can also be used to establish a severity code for system messages to allow the system to select default options for system messages. The NOHALT OCL statement does not override the halt mode established by the HALT command. Command halt mode is always prevalent (see chart below).

The NOHALT OCL statement changes the halt mode for the partition in which it was received. A NOHALT OCL statement takes effect immediately and lasts until a HALT OCL statement is received, another NOHALT OCL statement modifies the severity code, IPL is performed, or end of job occurs. (At IPL the system defaults to nohalt mode, with no severity code.) When the nohalt mode is reset at end of job, the system returns to the IPL nohalt mode or the mode established by the last HALT or NOHALT command received by the partition. The following chart shows the conditions resulting when both OCL and commands are used:

	NOHALT OCL	**HALT OCL**
NOHALT command	NOHALT	HALT
HALT command	HALT[1]	HALT

[1] Halts for end of job messages, end of job step messages, and system messages with no severity defaults or with severity codes greater than the current severity setting.

For a summary of Operator Control Commands, see Appendix C.

Placement

A NOHALT statement can be placed anywhere among the OCL statements. If a procedure it must precede the RUN statement.

Format

// NOHALT SEVERITY-code

Contents

The NOHALT statement, with or without the SEVERITY parameter, suppresses the message at end of job step and/or end of job.

SEVERITY: This parameter is used to indicate the severity code of messages that the system is allowed to select default options for. If the SEVERITY parameter is not specified, the operator must respond to all system messages, except informational, end of job step and/or end of job messages. The code must be one of the following:

1, 2, 4, 8 — The system selects the default option for system messages of a severity less than or equal to the code indicated. Severity codes and defaults are assigned to most system messages and cannot be altered. If the severity assigned to a system message is greater than the severity indicated in the NOHALT statement, the system stops, waiting for your response. If the severity assigned to the message is equal to or less than the severity indicated in the NOHALT statement, the system selects the default option for the system message and processing continues. If the default option selected by the system is a 2 or a 3, the end of job message is suppressed. The severity code does not affect system messages having no default options or requiring operator intervention. Severity code 1 is the least severe, severity code 8 is the most severe. Severity codes are reset to no severity at end of job. For more information on system messages and severity codes, see *IBM System/3 Model 15 System Messages,* GC21-5076.

Note: Halts displayed in the message display unit are not affected by the SEVERITY parameter.

Spooling Considerations

None

PAUSE Statement

Function

The PAUSE statement causes a system message (message 90 if before a LOAD or CALL statement; message 91 if between a LOAD or CALL and a RUN statement). It is usually used to give the operator time to prepare for the next program; for example, mount a different data module or insert special forms into the printer. Comment statements that give the operator instructions usually precede PAUSE statements (see *Comment Statement*).

When the operator is ready, the partition in which the PAUSE statement was received is restarted by responding to the message. The system then continues reading the OCL statements that follow the PAUSE statement.

The PAUSE statement is the only OCL statement logged on the display screen; therefore, it may be used to supply information to the operator.

Placement

PAUSE statements can be placed anywhere among the OCL statements. In a procedure, a PAUSE statement must precede the RUN statement.

Format

```
// PAUSE
```

Contents

None (comments may be entered after the blank following PAUSE)

Spooling Considerations

None

PRINTER Statement

Function

During system generation, a system print device is established for each partition. The PRINTER statement enables you to change the system print device for the partition in which the statement is processed. The new system print device remains in effect until changed by another PRINTER statement or until another IPL occurs.

Placement

The PRINTER statement can be placed anywhere among the OCL statements. In a procedure it must precede the RUN statement.

Format

// PRINTER parameters

Contents

The parameters are as follows (keywords are in capital letters; defaults are underlined):

DEVICE-{1403 / 3284}

LINES-nnn

FORMSNO-nnn

COPIES-nn

DEFER-{YES / NO}

ALIGN-{YES / NO}

PRIORITY-{0 / 1 / 2 / 3 / 4 / 5}

CLOSE-{YES / NO}

QCOPY-{YES / NO}

DEVICE: The device parameter is optional. If not specified, 1403 is assumed. Spool does not support the 3284.

LINES: The LINES parameter is optional. It is used to alter the number of print lines (forms length) per page. Possible range is 12 to 112. The number of lines specified remains in effect for that partition until another PRINTER statement with LINES parameter is entered or until the next IPL. This parameter overrides the forms length specified during system generation; however, a program's forms length overrides the LINES parameter. If a program's forms length is used, it is in effect for the duration of that job step only. At the end of the job step, forms length is restored to the previous value.

FORMSNO: This optional parameter, which applies only to the job step in which the PRINTER statement was used, may be used to inform the operator which forms are to be mounted on the printer. This parameter can be any combination of 1 to 3 characters, except commas, apostrophes, or blanks. When this parameter is used, the system halts with a message to the operator indicating the forms type to be used. When printing spooled printed output, the print writer issues a message whenever the forms type for the current print step is different from that of the previous print step. The response taken to this message informs the print writer if separator pages should be printed between jobs and job steps. See the *IBM System/3 Model 15 User's Guide to Spooling,* GC21-7632, for information on separator pages. If the forms type operand is given in the START PRT command, the print writer will print only those job steps whose forms type matches that given in the START command. You can start the print writer with a different forms type by entering the STOP PRT command, followed by the START PRT command with a forms type operand. Starting the print writer without the forms type operand causes the print writer to print job steps in the order they appear on the print queue.

COPIES: This optional parameter, which applies only to the job step in which the PRINTER statement was used, allows the user to obtain 1—99 copies of a job step's spooled printed output. If this parameter is not specified, only one copy is printed. When more than one copy is requested, the print writer continues to produce the number of requested copies before continuing to the next step on the print queue. All copies produced by the print writer allow for forms alignment if ALIGN-YES is specified. This parameter is ignored when print spooling is inactive or not supported for the specified device.

DEFER: The DEFER parameter is optional. It is ignored when print spooling is inactive or not supported for the specified device. DEFER-NO allows the spooling user to begin printing a job step's printed output before the job step has completed execution if the job step is the next step to be printed from the print queue. When DEFER-YES is specified, printing does not begin until the job step has completed execution. The DEFER parameter applies only to the job step in which the PRINTER statement was received. If the parameter is not specified, the system assumes DEFER-YES.

ALIGN: The ALIGN parameter is optional. It aids the operator in forms alignment for spooled printed output. This parameter is ignored when print spooling is inactive or not supported for the specified device. When ALIGN-YES is specified, the printer stops after printing the first line to allow forms alignment. A message is displayed on the CRT after the first line is printed. The operator's response to this message indicates that forms are aligned (continue printing) or that the line should be printed again (try alignment again). If more than one copy is requested (COPIES parameter) and ALIGN-YES is specified, the printer halts for forms alignment prior to printing each copy. If ALIGN-NO is specified, the printer does not stop. The ALIGN parameter applies only to the job step in which the PRINTER statement was received. If the parameter is not specified, the system assumes ALIGN-NO.

Note: If logging was assigned to the 1403, forms alignment is done on the first OCL statement logged to the 1403 for that job step. For this reason, logging to the 1403 should be suppressed when ALIGN-YES is used.

PRIORITY: The PRIORITY parameter is optional. A priority can be assigned to a job step to indicate its order of printing on the print queue. Priority 0 indicates a priority 1 job step that is to be held on the print queue until released via an operator control command. (See Appendix C for a summary of operator control commands.) Priority 5 is the highest priority that can be assigned. If this parameter is not specified, the priority of the job at the time of execution is assigned to the job steps on the print queue. This parameter is ignored when print spooling is inactive or not supported for the specified device. The PRIORITY parameter applies only to the job step in which the PRINTER statement was used.

CLOSE: This optional parameter is used to group multiple print steps of a job under one stepname. If CLOSE-NO is specified, no print steps are closed at end of step, only at end of job. If CLOSE-YES is specified, a print step is closed at end of step. If the parameter is not specified, CLOSE-YES is assumed. The CLOSE parameter is ignored when print spooling is not active.

If a previous print step specified CLOSE-NO, a PRINTER statement with an ALIGN, COPIES, DEFER, FORMSNO, PRIORITY, or QCOPY parameter is ignored and causes an informational message to be issued.

QCOPY: This optional parameter (QCOPY-NO) is used to prevent the spool file copy program from copying a job step that is on the print queue. QCOPY-YES allows the spool file copy program to copy the job step. If the parameter is not specified, QCOPY-YES is assumed.

The QCOPY parameter applies only to the job step in which the PRINT statement is included.

Spooling Considerations

You can change your system print device in a spooled job stream; however, if the new device is the 3284, printed output for the system print device (for example, from the overlay linkage editor or library maintenance program) is not placed on the print queue. To resume spooling of this printed output, you must change the system print device back to the 1403.

When a PRINTER statement is encountered and printer output for the job step is being spooled, the effect of the COPIES, DEFER, ALIGN and/or FORMSNO parameters is delayed until the print writer is ready to print the output.

PUNCH Statement

Function

The PUNCH statement enables you to define the system punch device for the partition in which the statement was received.

The new system punch device remains in effect until changed by another PUNCH statement or until another IPL occurs.

During system generation of a default system punch device is established for each partition.

Placement

The PUNCH statement can be placed anywhere among the OCL statements. In a procedure, it must precede the RUN statement.

Format

// PUNCH keyword parameters

Contents

The keyword parameters are as follows:

DEVICE-device

CARDNO-nnn

COPIES-nn

DEFER-{YES / NO}

QCOPY-{YES / NO}

PRIORITY-{0 / 1 / 2 / 3 / 4 / 5}

DEVICE: This parameter is optional and can be any of the following:

MFCU1	Primary hopper of the 5424
MFCU2	Secondary hopper of the 5424
1442	1442 Card Read Punch
MFCM1	Primary hopper of the 2560
MFCM2	Secondary hopper of the 2560
3741	3741 Data Station/Programmable Work Station (not supported for punch spooling)

CARDNO: This parameter is optional and may be used to inform the operator which card type to use for output punching. When the punched output is spooled, the card type used for punching can be identified in the PUNCH statement. This parameter may be any combination of 1 to 3 characters, except commas, apostrophes, or blanks. The CARDNO parameter applies only to the job step in which the PUNCH statement was used.

When the CARDNO parameter is used, the system halts and issues a message on the CRT indicating the card type to be used for the job or job step.

When punching spooled output on a 1442 Card Read Punch or 2560 MFCM, blank cards may be inserted between decks to cause all cards of the deck just punched to be fed into the stacker. On the MFCU, the last card is stacked without the extra blank cards.

COPIES: This optional parameter allows the user to obtain from 1 to 99 copies of job step's spooled punch output. If this parameter is not specified, only one copy is punched. When more than one copy is requested, the END OF PCH STEP message must be responded to before the punch writer will start punching the next copy. Once started, the punch writer punches the entire number of requested copies before continuing to the next job step on the punch queue. The COPIES parameter is ignored when punch spooling is inactive or not supported for the specified device. The COPIES parameter applies only to the job step in which the PUNCH statement was used.

DEFER: The DEFER parameter is optional. It is ignored when punch spooling is inactive or not supported for the specified device. DEFER-NO allows the spooling user to begin punching a job step's punched output before the job step has completed execution (if the job step is the next step to be punched from the punch queue). When DEFER-YES is specified, punching will not begin until the job step has completed execution. The DEFER parameter applies only to the job step in which the PUNCH statement was received. If the parameter is not specified, the system assumes DEFER-YES.

QCOPY: This optional parameter (QCOPY-NO) is used to prevent the spool file copy program from copying a job step that is on the punch queue. QCOPY-YES allows the spool file copy program to copy the job step. If the parameter is not specified, QCOPY-YES is assumed.

The QCOPY parameter applies only to the job step in which the PUNCH statement is included.

PRIORITY: The PRIORITY parameter is optional. A priority can be assigned to a job step to indicate its order of punching on the punch queue. Priority 0 indicates a priority 1 job step that is to be held on the punch queue until released via an operator control command. (See Appendix C for a summary of operator control commands.) Priority 5 is the highest priority that can be assigned. If this parameter is not specified, the priority of the job at the time of execution is assigned to the job steps on the punch queue. This parameter is ignored when the punch spooling is inactive or not supported for the specified device. The PRIORITY parameter applies only to the job step in which the PUNCH statement was used.

Example

```
1   4   8   12  16  20  24  28  32  36  40  44  48
// PUNCH DEVICE-MFCU2,CARDNO-50
```

In this example, prior to punching, the system halts and displays on the display screen, the partition's CARDNO parameter (50) informing the operator that card type 50 is to be placed in the secondary hopper of the MFCU.

During punching, no check for blank cards is made. If the system punch device is also the partition's system input device and neither device is being used by spooling, a halt is issued to inform the operator that the system punch device should be readied for punching.

Spooling Considerations

You can change your system punch device in a spooled job stream; however, if the new system punch device is not the spooled punch device, punched output from the overlay linkage editor and the library maintenance program, for example, is not placed on the punch queue. To resume spooling of this punched output, you must change the system punch device to the device designated as the spooled punch device during system generation.

The punch writer issues a message whenever the card type (CARDNO) for the current punch step is different from that of the previous punch step.

When a PUNCH statement is encountered and punch output for the job step is being spooled, the effect of the COPIES, DEFER and/or CARDNO parameters is delayed until the punch writer is ready to punch the output.

READER Statement

Function

The device used to read OCL statements is called the system input device. A default system input device is established for each partition during system generation. You can use a READER statement if you want to change your system input device.

Placement

The READER statement must not come between the LOAD and RUN or CALL and RUN. If you use the READER statement in a procedure, the system input device is changed when the statement is processed, but, OCL statements are not read from the new system input device until the procedure is completely executed. If you use the READER statement to change the system input device, the device you specify is used to read source programs, control statements, and OCL statements. Therefore, changing the system input device affects the placement of source programs and control statements as well as OCL statements.

The READER statement must be read from the current system input device or a procedure.

Format

```
// READER code (system input device)
```

Contents

Codes for the system input device can be as follows:

Code	Meaning
CONSOLE	CRT/keyboard (can be shared by partitions)
MFCU1	Primary hopper of the 5424
MFCU2	Secondary hopper of the 5424
MFCM1	Primary hopper of the 2560
MFCM2	Secondary hopper of the 2560
1442	1442 Card Read Punch
2501	2501 Card Reader
3741	3741 Data Station/Programmable Work Station

Spooling Considerations

You may change your system input device in a spooled job stream; however, if the new system input device is different from the spooled input device, jobs are not scheduled for execution from the reader queue. To resume scheduling jobs for execution from the reader queue, you must change the new system input device back to the system input device that is accepting spooled input.

RUN Statement

Function	The RUN statement indicates the end of the OCL statements for a job step. After the system reads the RUN statement, it executes the program specified on the LOAD statement or calls the procedure specified on the CALL statement.
Placement	A RUN statement is needed for each of the programs you want the system to run. In the job stream, it must be the last statement within each of the sets of OCL statements for your job steps. It can also be the last OCL statement in a procedure. (For more information about procedures, see *Procedures* in Part 2.)
Format	// RUN
Contents	None (comments may be entered after the blank following RUN)
Spooling Considerations	None

SWITCH Statement

Function

The SWITCH statement allows you to modify the external indicators for the partition in which the statement was received. Eight external indicators are assigned to each partition. External indicators allow you to influence the execution of your programs from an external source. The eight external indicators, for each partition, are set off during IPL. If a SWITCH statement sets an indicator on in a partition, it remains on until one of the following occurs:

- Another SWITCH statement sets it off.
- A JOB statement is received in the partition.
- A /. statement is received in the partition, in job mode.
- An IPL is performed again.

Placement

The SWITCH statement can be placed anywhere among the OCL statements. Only one SWITCH statement is allowed between the LOAD or CALL and RUN statements.

Format

// SWITCH indicator-settings

Contents

Indicator-settings: The indicator-settings parameter is a code that consists of 8 characters, one for each of the eight external indicators. The first, or leftmost, character gives the setting of external indicator 1; the second character gives the setting of external indicator 2; and so on.

The code must always contain 8 characters. For each external indicator, one of the following characters must be used:

Character	Meaning
0	Set the external indicator off.
1	Set the external indicator on.
X	Leave the external indicator as it is.

Example

The code 1X0110XX would cause the following results:

External Indicator	Result
1	Set on
2	Unaffected
3	Set off
4	Set on
5	Set on
6	Set off
7	Unaffected
8	Unaffected

Spooling Considerations

None

/& Statement

Function	/& statements are used as a precautionary measure. Placed at the end of your OCL for a job step, a /& statement signals the system that OCL statements for a new job step are coming. It prevents OCL for the next job step from being read as a part of the preceding set of statements or data. Any attempt to read more data from that device will be blocked. This statement terminates a step mode flush.
Placement	/& statements are not required. It is recommended, however, that you use them as the last statement in each of the sets of OCL statements for your programs. They are not allowed in a procedure.
Format	/&
Contents	None (Comments may be entered starting in column 3; however, this statement requires special consideration when used with the copy/dump program ($COPY). For more information regarding these special considerations, refer to *Card Input Considerations,* under *Copy/Dump Program*).
Spooling Considerations	None

/. Statement

Function

This statement is a job stream delimiter that has the following three functions:

A /. acts as a delimiter between jobs. It causes end of job and prevents the reading of more OCL for the job.

With spooling active, two consecutive /. statements indicate the end of the spooled input job stream.

With spooling inactive, this statement can be used to end job mode and return to step mode. It causes end of job in the partition in which the statement was received.

Placement

How this statement is used determines where it is placed in a job stream (see examples). It is not allowed in a procedure.

Format

```
/.
```

Contents

None (comments may begin in position 3.)

Spooling Considerations

Two consecutive /. statements must be used to indicate the end of the spooled input job stream.

A /. statement may be used to delimit jobs in a job stream. When a job being placed in the reader queue is not delimited by a /. statement, the input spooling routine generates a /. statement as the last statement for the job in the queue. This statement may contain extraneous characters following the /. when the system history area is displayed or printed, or when the statement is logged on the log device for the partition.

Some system service programs ($COPY for example) do not recognize a /. statement with comments or extraneous characters as an end of file indicator unless the statement is read from the system input device. Also, because /. statements in the reader queue may have extraneous characters following the /. delimiter, these programs should not attempt to read data from the reader queue unless the spooled reader is also the system input device for the partition in which that program is executing.

The following examples show how the /. statement can be used.

1. When used as a delimiter between jobs:

```
//JOB1 JOB
//STEPA LOAD PROGA,R1

// RUN
Data
/*
/&
//STEPB LOAD PROGB,R1
// RUN
Data
/*
/&
/.      This indicates the end of JOB1 and prevents the reading of more OCL for
        JOB1. If any job steps in JOB1 had been canceled, the /. indicates the end of
        the job. If this statement were not in a job stream, the following invalid JOB
        statement, the LOAD statement, and all other statements up to the next /. or
        valid JOB statement would have been read as part of the OCL for JOB1.
//JOB2JOB
//STEPA LOAD PROGC,R1
// RUN
```

2. When used to indicate end of spooled data:

```
//JOB1 JOB
//STEPA LOAD PROGA,R1
   (
// RUN
Data
/*
/&
//STEPB LOAD PROGB,R1
   (
// RUN
Data
/*
/.
//JOB2 JOB
//STEPA LOAD PROGC,R1
// RUN
Data
/*
/&
//STEPB LOAD PROGD,R1
   (
// RUN
Data
/*
/.  }
/.  } Indicates end of spooled input
```

3. When used to end job mode and return to step mode (nonspooled systems only):

```
//JOB1 JOB SPOOL-NO,PARTITION-1,CORE-12
//STEPA LOAD PROGA,R1
   (
// RUN
Data
/*
/&
//STEPB LOAD PROGB,R1
   (
// RUN
Data
/*
/.     Ends job mode.  Begin step mode.
// LOAD PROGC,R1
// RUN
```

*(Comment) Statements

Function

Comment statements are commonly used to explain the jobs or give the operator instructions. A comment statement can also be used to cause a second record to be written to the SHA following the comment. This record, called a time stamp, contains the system date and the time of day (if timer support was generated). Comment statements are printed along with the other OCL statements.

Placement

You can include, among OCL statements, special statements that contain only comments. Comment statements must contain an asterisk (*) in column 1. If you include the word TIME with an * (*TIME), the system writes the comment statement plus a time stamp in the system history area.

Comment statements can be placed anywhere among the OCL statements in either a job stream or a procedure. Comment statements are never displayed on the CRT but are printed if LOG is assigned to a printer (1403 or 3284) for the partition in which the comment statement was used and the printer is not allocated to a partition.

Format

* comment, or *TIME comment

Contents

The comment can be any combination of words and characters. The requirements are that the asterisk (*) be in column 1, and if specified, TIME must start in column 2.

The following example shows the format of the *TIME statement and the time-stamp record as it appears in the SHA.

```
(1) 2 *TIME COMMENT FROM PARTITION 2
(2) 2      04/26/78 00.00.54
```

(1) This statement was generated by the user.

(2) This statement was inserted into the SHA by the system.

Spooling Considerations

None

/* Statement

Function	The /* statement is not a true OCL statement but is used to indicate the end of a data file.
Placement	One /* should be used as the last card of an input data file or program deck. With the exception of card utilities, two consecutive /* statements will cause an error message.
Format	/*
Contents	None (Comments may be entered starting in column 3; however, this statement requires special consideration when used with the copy/dump program ($COPY). For more information regarding these special considerations, refer to *Card Input Considerations,* under *Copy/Dump Program.*)
Spooling Considerations	None

Part 2. System Concepts and Facilities

SYSTEM/3 MODEL 15 PROGRAMMING SUPPORT

System/3 Model 15 programming support includes system control programming (SCP) and program products (PP). These facilities allow a user to prepare and maintain disks and tapes, and perform basic functions necessary for the operation of a system.

SCP consists of disk system management programs and system control and service programs that are fundamental to the operation and maintenance of the system. Disk system management provides its support through the following:

- Initial Program Loader — Starts operation of the system by loading the supervisor into storage.
- Supervisor — Controls overall system operations and provides general function required by the scheduler and all processing programs.
- Scheduler — Initiates the execution of each new program and establishes the system facilities which are to be evoked while that program is running.
- Spool — Reduces processing unit dependence on the relatively slow speeds of unit record input/output devices, and reduces contention for the devices.
- Data Management — Provides routines to interface between a user program and the required data files. Interfaces are provided for files on disk, tape, cards, or diskette, and for the printer, CRT, BSCA, or SIOC; also, a device-independent access method is supported.

System control and service programs enable the user to service the program libraries, data files, application programs, and input/output units. Following are some of the programs included in this group:

- Library Maintenance — Allows the user to produce, maintain, and service the source and object program libraries
- Copy/Dump — Supports file-to-file and volume-to-volume copies
- File and Volume Label Display — Displays information about the contents of a disk
- File Delete — Deletes data files from a disk

The following program products are available to satisfy specific application requirements:

- RPG II
- Subset ANS COBOL
- FORTRAN IV
- Basic Assembler
- Disk Sort
- Tape Sort
- CCP/Disk Sort
- Card Utilities

PROGRAM CONCEPTS

Any set of user instructions for processing data must go through several phases before it can be used by the system to actually process data. User-written instructions form a source program; the source program is processed by a language translator to form an object module; the object module is formatted by the overlay linkage editor into a load module; the load module can be executed by the system. The following discussion describes these steps in greater detail.

Source Programs

A source program is a set of user instructions that can be compiled and used for processing data. To write a source program, a user must analyze the input data, decide what must be done to it, and determine the format of the output.

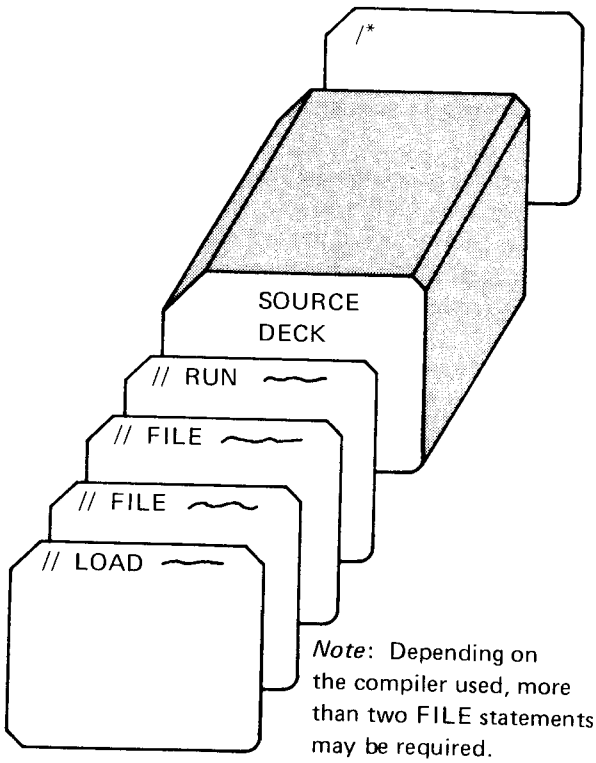
After this analysis, the user writes instructions according to the conventions of a programming language (such as RPG II) to process the data. These instructions taken together are called a source program and the user can punch it into cards, write it on a diskette, store it on disk prior to compilation or assembly, or enter it into the system directly from the console/keyboard.

Object Modules

An object module is a source program converted into instructions that can be link-edited. To obtain an object module, a source program is processed by a compiler (such as the RPG II compiler) or an assembler. The resulting object module contains the necessary machine instructions required to perform the desired processing of data. From the compiler or assembler, the object module can be stored on disk, punched into cards, or written on a diskette.

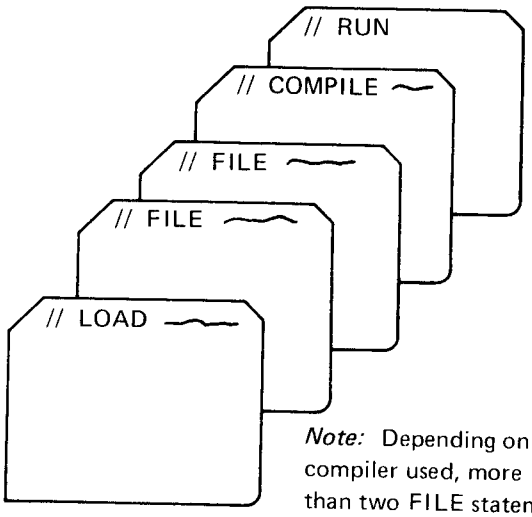
The following examples show typical OCL used to compile source programs.

Job stream for compiling a source program punched in cards.



Note: Depending on the compiler used, more than two FILE statements may be required.

Job stream for compiling a source program located on disk in a source library.

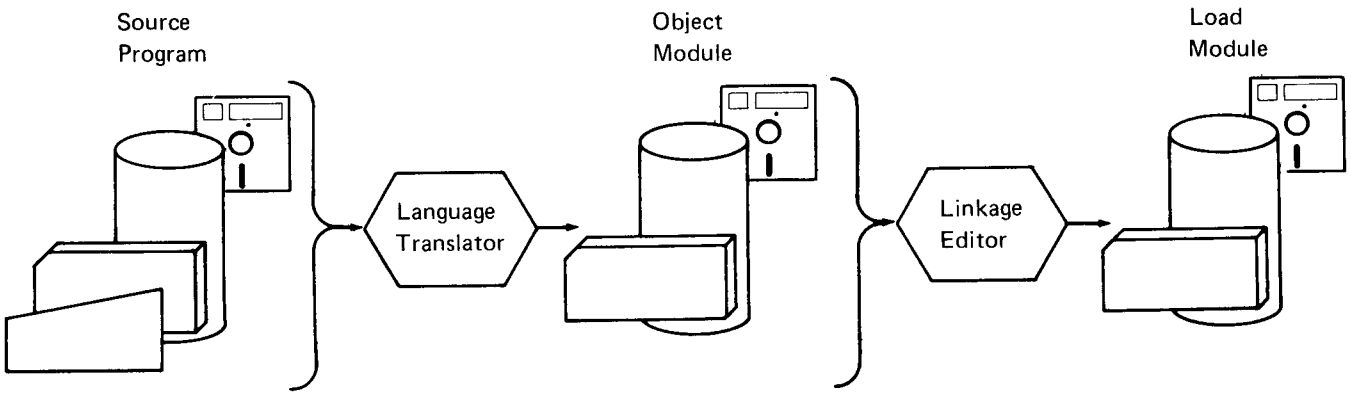


Note: Depending on the compiler used, more than two FILE statements may be required.

Load Modules

A load module consists of at least one object module that has been changed by the overlay linkage editor into a module that can be loaded for execution.

Linkage editor processing is necessary following the assembly or compilation of any program. The output of a language translator (assembler or compiler), called an object module, cannot be run as a program until it is link-edited into a load module. Object modules and load modules can reside on cards, on diskette, or in an object library on disk.



Overlay Linkage Editor

The overlay linkage editor provides services to the language translators. The following section provides an overview of these services; for detailed information on the overlay linkage editor and its uses, see *IBM System/3 Overlay Linkage Editor Reference Manual*, GC21-7561.

The overlay linkage editor can be requested by a user or directly by a language translator such as FORTRAN, COBOL, RPG II, and Basic Assembler.

The following OCL statements show how to load the overlay linkage editor:

```
// LOAD $OLINK,unit
// FILE NAME-$SOURCE, . . .
// FILE NAME-$WORK, . . .
// RUN
```

(These two FILE statements, which are optional, are standard FILE statements used by the compilers.)

The overlay linkage editor provides the following functions for the language translators:

- Punches the object module into cards, writes it to a diskette, and/or catalogs it into an object library on disk. These modules are also referred to as R modules, routines, or nonexecutable object programs. They are programs and/or subroutines that still need to be link-edited into load modules.
- Link-edits the object module(s) into a load module; punches this load module into cards, writes it on diskette, and/or catalogs it into an object library on disk. These modules are also referred to as O modules or object programs. They are programs and/or subroutines that can be loaded for execution.

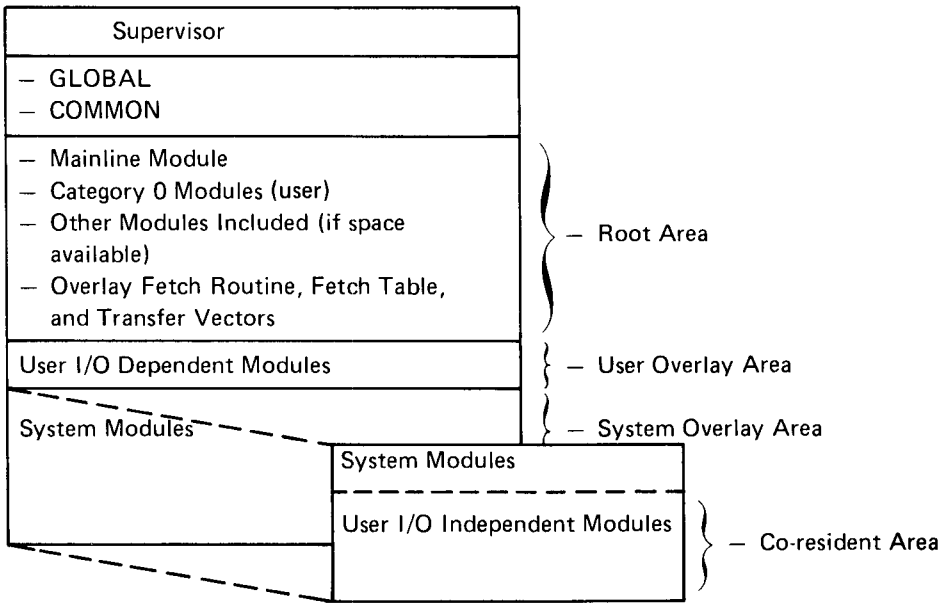
When the overlay linkage editor link-edits one or more object modules, it attempts to fit the resulting load module into the user-specified program size or the current program partition size. If this cannot be done, the overlay linkage editor assigns some modules to overlay segments. Main storage for an object program with overlays is divided into four areas: root area, user overlay area, system area, and co-resident area. Not all programs need all four areas.

The root area of an overlay program contains the parts of the program that are never overlaid. The root area always contains the mainline module, overlay fetch routine, fetch table, and transfer vectors. The remaining parts of the root area depend on the program being linked.

The user overlay area contains user modules that call system I/O modules. Each overlay (known as a segment) loaded into the user overlay area can contain modules of different category values. Category values determine what modules reside in the various areas. If the COBOL segmentation feature has been used, the COBOL segments appear as overlays in the user overlay area. The presence of COBOL segments forces any non-COBOL modules that normally would have been assigned to the user area to the root area (category 0).

A system overlay segment contains system modules with the same category value. Each system overlay segment is independent of other system overlay segments. System modules are assigned to overlay segments solely by category value. A system module can call only another module with either the same category or a category 0 module. The co-resident area is actually a part of the system overlay area.

The following example shows overlay areas.



The maximum number of overlay segments in a program is 254. A storage map provided by the overlay linkage editor indicates overlay area addresses and the segments each overlay area can contain.

The following OCL statements are an example of a language translator requesting the overlay linkage editor:

```
// LOAD     $RPG, unit (unit can be R1, F1, R2,
            or F2)
// COMPILE  SOURCE-PROG1,UNIT-unit,
            OBJECT-unit,ATTR-MRO
// FILE     NAME-$SOURCE, . . .
// FILE     NAME-$WORK, . . .
// RUN
```

Memory Resident Overlays

Memory resident overlays is a technique designed to increase the performance of large overlay programs by allowing certain overlay segments to remain in main storage after the initial segment fetch. The two types of memory resident overlay programs are MOVE and REMAP, which differ as follows:

- When ATTR-MOVE (MOVE technique) is specified in the OPTIONS statement of the overlay linkage editor, the user program retains the segment in the resident area but executes the segment in the conventional overlay fetch area.
- When ATTR-MRO (REMAP technique) is specified in the OPTIONS statement of the overlay linkage editor, the program executes the segments in the resident area itself.

The ATTR parameter is to be used only with load modules; the MRO and MOV program attributes will not be attached to R modules (nonexecutable object programs).

For more information, see *IBM System/3 Overlay Linkage Editor Reference Manual*, GC21-7561.

To be used, the memory resident overlay feature must be selected as an option during system generation. However, load modules may be link-edited with this attribute on any system.

The overlay fetch routine generated for the MOVE technique is identical to the fetch routine generated for conventional overlay programs.

The CATEGORY statement controls which overlay segments are candidates for memory resident overlays. Any overlay segment containing a category 125 module is not a candidate for memory residence.

With the two memory resident overlay techniques (MOVE and REMAP), large programs can reside in primary storage throughout execution if the partition is large enough and if the program can be link-edited within the maximum program size. These techniques may improve performance for overlay programs that require a large number of overlay fetches.

The REMAP technique requires that the overlay segments be link-edited to 2K boundaries. The MOVE technique does not have this restriction. For large overlay segments, REMAP generally executes faster than MOVE.

Throughput degradation for memory resident overlay programs with RLDs (relocation dictionary records) is not as severe as for conventional overlay programs with RLDs, because each resident overlay segment is relocated only the first time it is fetched from disk.

PROGRAM AND PARTITION SIZES

Program size is the amount of main storage (excluding external buffer requirements) required for a program to execute. Partition size is the amount of main storage available for executing a program. Program and partition sizes are related items because the partition size must be large enough to accommodate the program. When the program size (plus external buffer requirements) is larger than the partition, either the partition size can be increased or the program can be structured with overlays.

The partition sizes that best meet the needs of an installation depend upon such factors as the total amount of storage available, the size and characteristics of the user programs, their balance among job streams, and the operating environment.

Partition sizes are specified in 2K increments; the minimum size in which to execute a program is 8K. There is no upper limit, except as determined by the system. The maximum program size is 48–56K. A program to execute under CCP can range from 4K to 32K. The maximum program size is dependent on the system configuration chosen during system generation. Generally, the more system generation options selected, the smaller the maximum program size. For example, the maximum configuration that also includes CCP allows only 48K programs; whereas the minimum system configuration, without CCP, supports 56K program size. Also, canceling spool can potentially increase the maximum allowed program size by one 2K increment. The increased size allows more programs to run without overlays and, therefore, may reduce execution time.

Since partition sizes are specified in 2K increments, care must be taken when a program is changed. A program size exceeding a 2K boundary by as little as 1 byte requires the partition size to increase another 2K. See the following example:

	Program Size	Partition Size
Original program	8,100 bytes	8K
Changed program	8,193 bytes	10K

The partition size is initially set during system generation; later it may be reset with a SET command. The program size can be specified by the user either when writing a program (via programming language specifications) or when link-editing the program (via the CORE parameter on the OPTIONS statement).

The DISPLAY STATUS command provides system information on the CRT. Included in this information are the partition sizes. The display also provides the maximum program size that can be executed on that particular system.

Greater Than 48K Programs

DISPLAY STATUS on the system display screen specifies the maximum program size available (MAX PROG SIZE = XXK). Possible values are 48K, 50K, 52K, 54K, or 56K depending on the options chosen during system generation. The user can direct the overlay linkage editor to generate a load module with a specified maximum limit; this information (object core size) is specified as follows:

RPG	RPG control card (H)
FORTRAN	CORE statement
COBOL	OBJECT-COMPUTER paragraph
$OLINK	OPTIONS control statement

The value specified should never be larger than the maximum allowable program size (48K-56K). If the desired execution size is not specified, the overlay linkage editor assumes the current partition size (used for compilation/assembly) or 48K, whichever is less.

A link-edit address of X'4000' should be specified in the COMPILE statement (LINKADD) for all programs to be executed in a partition (this is the compiler's default value). If the program is to run as a CCP task, X'8000' should be specified. (Maximum program size for CCP tasks is 32K.)

EXTERNAL BUFFERS

An external buffer is an area located outside the user program but within the user partition. An external buffer contains the disk buffers for the user program. When a user program requests external buffers during compilation, the disk buffers are moved out of the user program area and located after the last user program byte. These disk buffers are now located between the end of the program and the end of the partition. Processing of data in these buffers is done in *move mode*. This removal of disk buffers from the root areas of a program allows more executable code to be included in the root area. This can have the effect of increasing the maximum program size.

External buffers are supported by System/3 COBOL, FORTRAN, RPG II, and Basic Assembler. External buffers are requested as follows:

- COBOL — Two parameters, EXTBUF and NOEXTBUF, are supported on the PROCESS statement. EXTBUF provides external buffers for all disk files; NOEXTBUF provides buffers within the program. NOEXTBUF is the default. For further considerations in using the SAMEAREA clause with EXTBUF, see *IBM System/3 Subset ANS COBOL Reference Manual,* GC28-6452.

- FORTRAN — Two parameters, EXTBUF and NOEXTBUF, are supported on the *PROCESS statement. EXTBUF provides external buffers for all direct disk files; NOEXTBUF provides buffers within the program. NOEXTBUF is the default. For further considerations in using EXTBUF and SHRBUFF, see *IBM System/3 FORTRAN IV Reference Manual,* SC28-6874.

- RPG II — An E is specified in column 48 of the Header Specification to provide external buffers for all disk files; not specifying an E in column 48 provides buffers within the program. The default is no external buffers.

- Basic Assembler — The parameters, XBUF-n and NOXBUF, are supported on the OPTIONS statement; n is a one- to five-digit decimal number specifying the size of the external buffer required. NOXBUF is the default.

External buffers are intended for very large programs that previously required a severe reduction in the disk buffer sizes to enable programs to fit within a partition. Specifying external buffers for these programs can improve performance because larger buffers can be utilized. Small programs, however, may experience performance degradation due to the extra processing required for external buffers. The maximum size for external buffers is 64K. Therefore, as an example, a 56K program plus 64K of external buffers would require a partition size of 120K.

File Facilities

FILE DEFINITION

A file is a collection of related records to be treated as a unit and contained on cards, disk, diskette, tape, printer, BSCA, or SIOC.

This chapter describes file organization and processing in general. These subjects are discussed in greater detail in the *IBM System/3 Disk Concepts and Planning Guide,* GC21-7571, and *IBM System/3 Magnetic Tape Program Planning Manual,* GC21-5040. For more information on file processing with respect to programming languages, see *IBM System/3 RPG II Reference Manual,* SC21-7504, *IBM System/3 ANS COBOL Reference Manual,* GC28-6452, and *IBM System/3 FORTRAN IV Reference Manual,* SC28-6874.

File Organization

Three types of file organizations are defined based on the arrangement of the records within a file: sequential, indexed, and direct.

A *sequential file* is a file in which the position of a record is determined by the order in which records are put in the file. For example, the tenth record put in the file occupies the tenth record position. Files on cards, diskette, and tape are always sequential files; disk files may be sequential files.

An *indexed file* is a disk file in which the location of a record is stored in a separate but adjacent portion of the file called an index. The index has a record key and record location for every record contained in the file. An index enables a program to process only required records; it is not necessary to access all the records of the file.

Note: Indexed processing is not allowed on the simulation areas of the 3340 and 3344.

A *direct file* is a type of sequential disk file in which records are assigned specific record positions by the user. Direct file organization enables accessing any record in the file without examining other records or searching an index. Records are assigned specific locations, independent of the order they are put into the file. A user defined control field on the record determines the record's specific location in the file. Therefore, records can actually be scattered throughout the file, depending on the control field. Unused record locations contain blanks.

File Processing

Files can be processed by three basic methods: consecutive, sequential, and random.

The *consecutive* method processes records in the order in which they physically appear in a file. The consecutive method can be used for sequential, indexed, and direct files. The contents of spaces left for missing records in direct files are read as blank records. Records are read until either the end of the file is reached or the program terminates the reading of records.

Sequential processing applies only to indexed files. When an indexed file is processed sequentially, the record keys are processed one after another in ascending order. If the records are not in order in the file, they can be processed in order by means of the keys in the file index. There are two ways to sequentially process an indexed file: by key and within limits. Sequential-by-key processes all records in the order of their key fields. Processing continues until all records have been read or the program terminates the processing. Sequential-within-limits allows a section of the file (group of records) to be processed. Each section is identified by lower limit (starting) and upper limit (ending) record keys.

Note: COBOL supports lower limit processing only; the upper limit must be provided within the COBOL program.

Random processing allows disk records to be processed in an ordered or unordered manner; a particular record can be processed independently of its relation to other records. Sequential and direct files can be processed with a relative record number to identify the record. The relative record number indicates the position of the record within the file in relation to the beginning of the file. It is not a disk address, but a positive, whole number that is converted by disk data management to the disk address of the record.

The relative record numbers can be contained in an ADDROUT file (record address file) created by the Disk Sort program. ADDROUT files are comprised of binary 3-byte relative record numbers that indicate the relative position of records in the file to be processed. To process indexed files randomly, the user must use the record keys in the file index to identify the records.

File Creation

Data is placed in a file according to user specifications. When disk or tape files are created, a FILE statement for each file must be included in the OCL for the program.

When creating a disk file, the FILE statement supplies the name of the file, the retention of the file, the file size, the area (simulation or main data area) to contain the file, and (optionally) the location within the area.

The name given to a file is the name a program will use in referencing that file. Some programs require specific file names to be used, such as $WORK for compiler work files. (FILE statement descriptions in Part 1 contain a list of the reserved file names.) Several versions of a file can be created on the same disk and be given the same name, but the date must always be unique. Each version can be referenced by its location, size, or unique date.

The retention of a file is classified as scratch, temporary, or permanent. A scratch file may be used only in one job step and cannot be retrieved after that job step has ended. The first time the Model 15 allocates new space on a simulation area that has scratch files created by another System/3 model, all scratch files are removed. A message is issued before removal, allowing the job step to be terminated or continued.

A temporary file is usually used more than once; however, the space containing the file can be reused under one of the following conditions:

- A FILE statement specifying scratch is later supplied for the temporary file. This removes the file from the VTOC (volume table of contents).
- Another file with the same LABEL name is loaded into the exact area occupied by the temporary file; this only changes the data. Space and location parameters are required.
- The file delete program is used to delete the file.

The space containing a permanent file cannot be used for any other file until the file delete program has deleted the file. If the use of a file is not specified when created, the file is classified as temporary. A temporary file can be changed to a permanent file only if the file name is changed and it is copied as a permanent file.

The output file is scratched at end of job step (just as if RETAIN-S has been specified for the output FILE statement) when all of the following conditions exist:

- A pack containing an input file is not online at the start of the job (deferred mount).
- The output file is to be written over the input file (load to old).
- RETAIN-S is used on the FILE statement for the input file.

An existing temporary file should be reloaded (load to old) with files of like attributes. If an existing indexed file is reloaded with a sequential file, the new data overlays only the data portion of the file; the index portion of the file remains intact but unusable.

The amount of disk space for a file depends on the size of the records, the number of records (both current and to be added in the future), and the file organization.

The size of the file can be specified by either the number of tracks needed or the approximate number of records for the file. When the number of records is given, the system calculates the required disk space by converting number of records to number of tracks.

The total space allocated is rounded up to full tracks, allowing adequate space to accommodate at least the number of records indicated. This means the file could hold more records than specified, allowing additional records to be added to the file. Therefore, if the copy/dump program ($COPY) is used to copy the file to another disk, more records may have to be specified than were specified when the file was created.

File Location

After the size of a file has been determined, disk space to contain the file can be allocated. A main data area can contain up to 1000 files; a simulation area can contain up to 50 files. A location may be specified on the FILE statement indicating the beginning of the file: for a simulation area, it is specified via a track number; for a main data area, it is specified via cylinder/head. Allocation of space for a file on either a main data area or a simulation area begins at the specified location and extends toward the high cylinder end of the area.

Note: When using the COPYPACK function of $COPY and $DCOPY to copy an entire 3344 pack, files located on cylinders 167–186 of a 3344 main data area will be copied to another 3344 main data area or tape, but they will not be copied to a 3340 main data area.

The system requires a location to be specified on the FILE statement when creating a file with an identical label and the same size as a file that already exists in the area, when reloading over an existing file, or when loading an offline multivolume file to disks that contain other files. When a file is referenced, the location is used for a more specific identification check and for identifying one of several files having the same label and same size.

A device independent FILE statement for card, diskette, and printer files allows I/O devices to be assigned when a program is executed. Thus, a program need not be rewritten when a different device is to be used. (For example, input originally read from the MFCU1 can be read from the 3741 without any change in the file description specifications in the RPG II program.) The FILE statement supplies the system with information about I/O devices used in the program. This information is used to read records from and/or write records to the specified I/O device. A device independent FILE statement must be supplied for each device independent file used in a program. If device independent files use disk or tape units for input or output, then the disk or tape FILE statement is used.

A system service program ($FCOMP) is used to remove gaps between files. The program can reorganize the files on a specified main data area or copy (add) an entire main data area file by file to another main data area. LOCATION parameters may have to be changed in OCL statements after $FCOMP is used. (See *File Compress Program—$FCOMP* in Part 4 for more information.)

Automatic File Allocation

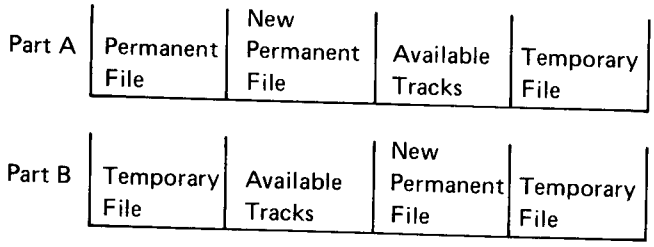
When the location of a new file is not specified on the FILE statement, the location is determined by disk system management. The process is known as automatic file allocation.

When allocating file space, disk system management calculates the length of the file and, for a simulation area, checks the volume label to determine which tracks are available for allocation. (The volume label contains the status of each track and indicates which tracks are available for use.) File space is first allocated for permanent files, then temporary files, and finally scratch files, if multiple files are being allocated.

Disk system management places the file on the smallest contiguous string of available tracks that can contain the file, leaving as few empty spaces as possible. For example, if the file is 10 tracks long and there is one string of 12 available tracks and another of 15 tracks, the file is placed in the string of 12 tracks because the 12-track string is closer to the length of the file.

If disk system management, while searching a *simulation area*, finds two strings having the same number of available tracks, the file is placed at the highest numbered available location. Also, if the file is the first file placed on a disk, the system allocates space for the file beginning at the highest numbered track. The system allocates space beginning at the highest location to allow as many available tracks as possible next to the object library (the libraries are usually located at the lowest numbered tracks) to enable the object library to expand if necessary.

If the area found for a new file contains more tracks than required, disk system management determines the type of file to the left (lower numbered track) of the available tracks. If the file to the left is of the same retain type, the new file is left-adjusted; if the file to the left is not similar, the new file is right-adjusted as shown in the following example:



Part A: The file is left-adjusted since both files are permanent.

Part B: The file is right-adjusted since one file is temporary and the other is permanent.

When disk system management is allocating space on a *main data area*, the search for space begins at cylinder 1 and extends toward the high cylinder end of the disk. When a file is classified as temporary or permanent, all disk space is searched to find the available space that best fits the file. The file is adjusted toward the cylinder 1 end of the available space. For a scratch file being placed on a 3340 main data area, the entire main data area is searched for the best fit; the file is adjusted toward the cylinder 166 end of the available space. For a scratch file being placed on a 3344 main data area, the disk space between cylinders 167–186 is searched for the best fit. If sufficient space exists, the file is adjusted toward the cylinder 186 end of the available space. If space cannot be found within cylinders 167–186, the entire main data area is searched for the best fit; the file is adjusted toward the cylinder 186 end of the available space.

Although it is easier to let disk system management allocate file space, the following are some advantages in the user determining file allocation. More efficient file locations may be determined by a user than by disk system management. Disk system management may leave a string of available tracks between files that is unusable because the string is not long enough to contain another file. The user can determine the location of all files by using the file and volume label display program.

Automatic file allocation considers effective use of file space, but not the usage of the files. It does not consider file planning for multiple input files in a program or job-to-job transitions. When a user plans file locations, files used together can be placed near one another on disk; thus, processing time may be improved.

A function known as auto-allocate provides automatic allocation of space for work files. This function can be used by system programs such as library maintenance and disk sort. Using the auto-allocate function of the disk sort program generally increases the time needed to run a sort job; auto-allocate does not always provide the work file arrangement needed for a fast sort run. When a user is concerned with minimizing sort run time, a work file should be specified by means of a FILE statement, rather than allowing the system to automatically allocate work space. An advantage of using auto-allocate with disk sort is, if sufficient contiguous space is not available, the system will find work space that may be located in different noncontiguous spaces of the same volume or on different volumes.

FILE SERVICES

The following system service programs are provided for servicing files:

- The file and volume label display program ($LABEL), has two uses:
  - Print the entire volume table of contents (VTOC) for an area (main data or simulation)
  - Print the VTOC information for only certain data files

  In both cases, the program also prints the name of the area.

  The printed VTOC information is a readable, up-to-date record of the contents of the area. This information may have a number of uses, such as:
  - Check the contents of an area to ensure that it contains no libraries, permanent data files, or temporary data files before reinitializing
  - Locate space that is available for libraries or new files
  - Obtain specific file information, such as the file name, designation (permanent or temporary), or the space reserved for the file
  - Determine the amount of space available in a particular file
  - Provide file location for use of the $COPY recovery function

  The control statements for the program depend on the program use. For more information see *File and Volume Label Display Program—$LABEL* in Part 4 of this manual.

- The copy/dump program ($COPY) performs only one of the following functions per execution:
  - Copy an entire volume
  - Copy a data file
  - Copy and print a data file
  - Copy a data file, but print only a part of the file
  - Print an entire data file
  - Print only a part of a data file
  - Print and copy a part of a data file
  - Build a direct file from any file type, except REORG-YES may not be specified for an indexed file
  - Build an indexed file from a sequential file
  - Recover a file
  - Copy a file with the output record length different than the input length

  The control statements used depend on the desired results. For more information, see *Copy/Dump Program—$COPY* in Part 4 of this manual.

- The file delete program ($DELET) can be used to:
  - Remove all files from the VTOC and optionally remove their associated data from the area.
  - Remove a specific file by name from the VTOC and optionally remove their associated data from the area.
  - Remove file references in the VTOC only. This frees the space they occupy for use by new files but does not remove the data from the area.
  - Free space that has been allocated but, due to abnormal circumstances, is not associated with a file or library.

- The recover index program ($RINDX) recovers indexed files by providing the following functions:
  - Recover records added to indexed files lost because of abnormal termination
  - Update the format-1 label in the scheduler work area with new end-of-data and end-of-index pointers
  - Call the system key sort program and update the VTOC at end of job

- The file compress program ($FCOMP) has two primary functions:
  - Place all files (except $SPOOL) in a main data area together at the cylinder one end of the disk
  - Copy each file in a main data area to another disk

SCHEDULER WORK AREA

The scheduler work area (SWA) is a work space located on the system pack. One use is to temporarily save file label (F1) information during the processing of a program. The library maintenance program is used to create a scheduler work area for each partition. Space for this area is assigned immediately preceding the object library.

The space for file label information is 48 sectors, and it can contain a maximum of 192 entries, each 64 bytes in length. A maximum of 192 entries (files, volumes of a multivolume file, or a combination) may be specified for one program. In some cases, however, the maximum will be less than 192. When all files are indexed multivolume files, the maximum is 96 files. A 10K partition must be available to process 192 entries; a partition of 8K can process up to 128 entries.

The auto-allocate function may require an F1 entry in the SWA even though it is not specified by the user. For example, the Disk Sort program has an auto-allocate function wherein the system, not the user, locates work space for the sort. Also, some of the system service programs use the auto-allocate routines.

Generally, one format-1 label is required for each file. One F1 label represents one FILE statement for disk, tape, or device independent data management (DIDM). For multivolume files, there is one F1 label for each PACK or REEL name. In addition, one F7 label is used for each volume of an indexed multivolume file to contain HIKEY information.

The following chart can be used to determine the number of SWA entries required for each program execution. A direct file requires the same number of entries as a sequential file.

Type of File	Number of Volumes	Applicable Devices	Number of SWA Entries
Sequential	1	Disk, Tape, DIDM[2]	1
Sequential (MVF)	3	Disk, Tape, DIDM[2]	3
Indexed (created as single volume)	1	Disk	1
Indexed (created as MVF)	1	Disk	2
Indexed (created as MVF)	3	Disk	6
Auto-allocate:			
$DSORT	Up to 4	Disk	Up to 4
$MAINT	1	Disk	1[1]

[1] Not including files specified for the file-to-library or library-to-file functions.
[2] Device independent data management.

For example, when the $COPY program is used to copy a multivolume indexed file on five volumes to another five volumes, 10 entries are required for the input file and 10 entries are required for the output file. Thus, by summing these requirements, it can be determined whether the maximum allowable number of SWA entries (192) has been exceeded.

FILE SHARING

File sharing allows a disk file to be shared by two or more programs executing at the same time. Programs can read from, update, or add to the file.

File sharing allows:

- Programs executing in batch mode in different partitions to input from, update, and add to the same file
- A program in a partition to access a record added by another program in a partition while both are executing. (For more information, see *Sharing Access To Added Records* chart under *File Share* in *Multiprogramming Considerations and Restrictions*.)
- A CCP-defined file to be processed by a batch program without requiring CCP to suspend processing or shutdown

Sharing a file reduces the need to have more than one copy of a file online, and it reduces file contention between programs. File sharing is at the block level (one block of records) rather than at the file level. Thus, two or more programs can be processing records from the same file, but simultaneous accesses to the same block are queued.

Compatible Access Methods for File Sharing

Different file types can be accessed by the different types of disk data management. For example, an indexed file can be accessed by consecutive input data management. The following chart shows the data management access methods allowed to access the different file types.

Access Method	File Type[1] Consecutive	Direct	Indexed
Consecutive	YES	YES[2]	YES[2]
Direct	YES	YES	YES
Indexed	NO	NO	YES

[1] Indicates the type of file organization used to create the file.
[2] Except consecutive add.

Figure 2-1 shows the compatible disk data management access methods allowed with file sharing. The access methods on the left indicate the data management accessing the file.

CAUTION:
If you use consecutive or direct data management to update an indexed file, you should exercise care to prevent destroying the key field portion of the record.

The data management access method is indicated by 2, 3, or 4 character codes as follows:

- The first position—C, D, or I—corresponds to the file organization—consecutive, direct, and indexed.
- The 2nd, 3rd, and 4th positions can contain these characters: R and S correspond to file access—random, sequential. L refers to processing sequentially within limits. O, A, U, G refer to function—output, add, update, and get.

Example:

1. A file is open as consecutive input (CG).
2. An attempt is made to open the same file with indexed random input, update, and add (IRUA). The file is allowed to open if it was created as an indexed file.

DTF (Define the File)

A DTF (define the file) is a file control block generated by the compiler. It resides in the object program, can vary in size from 30 to 150 bytes (depending on the file organization and access method), and is used by data management to communicate with the user program. At least one DTF exists for each file, including printer, tape, disk, card, 3741, console, or device independent files.

As part of opening files for program execution, the system's allocation and open routines store VTOC information about the file in the DTF. This information includes file name, device code, file attributes, pointers to input/output blocks, command code, and I/O completion code.

For file sharing (disk files only), the DTF also contains a pointer to the file share DTF (SDTF).

SDTF (Share Define the File)

When a disk file is being shared, an additional control block, called the SDTF (file share DTF), is used. It resides in main storage outside the program partition in a common area called the file share area. There is one SDTF for each physical, open, shared disk file; the SDTF is created when that file is opened the first time, and it is removed when the file is closed for the last time.

The SDTF contains information that reflects the status of the file at any particular moment. Within the SDTF are file attributes, addresses of disk data extents, a counter that indicates the number of times the file was opened, and a pointer to file share queue elements (FSQE). Data management uses the SDTF to obtain and update the status and pointers of shared files.

The SDTF varies in size depending on the type of disk file it describes. The basic SDTF size consists of a block of 64 bytes. SDTF requirements are:

	Number of SDTFs	Number of Blocks	Number of Bytes
Single volume file			
Sequential or direct	1	1	64
Indexed	1	2	128

	Number of SDTFs	Number of Blocks	Number of Bytes
Multivolume file			
Sequential or direct			
1 volume	1	2	128
2 volumes	2	4	256
3 volumes	3	6	384
4 volumes	4	8	512
Indexed			
1 volume	1	3	192
2 volumes	2	6	384
3 volumes	3	9	576
4 volumes	4	12	768

FSQE (File Share Queue Element)

An FSQE is used to record what portions of the file are currently locked (enqueued) and to indicate if contention for those locked portions exists.

The FSQE portion of the file share area is a work area used during disk file processing. An FSQE is created for each reference (for example, GET or PUT) to a physical, shared disk file that is opened for update or add. There can be more than one FSQE for each SDTF.

File Share Area

File sharing requires an area at the high end of main storage to contain the SDTFs, FSQEs, and a common area. The user specifies the size of the entire area during system generation. This size can later be changed with the SET command in increments of 2K (all partitions must be at end of job before the SET command can be used).

The file share area is specified as a multiple of 2K; 2K is required as a minimum. The system divides this area for storage of the SDTFs and FSQEs in a predetermined ratio. For example, in a 4K file share area, there is space for 52 SDTFs and 35 FSQEs.

If the user decides (through an analysis of his applications) that the file share area would be better utilized with a different ratio of SDTFs and FSQEs, the $CNFIG (configuration record) program can be used to assign a different proportion. (See the FSHARE statement in the $CNFIG system service program.)

Doubly-Defined Files

In a single user program, you can define one physical disk file with two different names and file descriptions. Such a doubly-defined file requires certain user conventions in order to avoid file contention problems.

File contention occurs when the same records in a file are being accessed for update by different users at the same time. Contention for a doubly-defined file occurs when the user simultaneously processes the same records.

Data management routines ensure that shared update files are protected during the update. As the file is being processed, the record to be updated is read into main storage. The disk block that contains the record, and any other records in the block, is then protected from any other update access. Requests for information in that block are locked; that is, the request waits until the updated record is written back to disk, at which time the original request is unlocked, and the new request can be processed.

(*Note:* For System/3, the lock/unlock mechanism is called enqueue/dequeue.)

During random processing, this rewrite/unlock/dequeue occurs after each update of the record. During sequential processing, this rewrite/unlock/dequeue does not occur until the entire block of records has been processed. In either case (and especially when processing sequentially), programs that use doubly-defined files are susceptible to lockouts.

For further explanation, consider the following example. Assume that an indexed file has been defined in one program with two different definitions:

File Name	Access Method
FILEA1	Direct update (DU)
FILEA2	Indexed random update and add (IRUA)

If the user has retrieved a record from the file using the FILEA2 (IRUA) definition, and then attempts to retrieve that record (or one that is in the same block) using the FILEA1 (DU) definition, the latter access will be locked out until the record is written back to disk for the FILEA2 definition.

If the sequence of events is such that the rewrite to FILEA2 does not occur until the successful completion of the read request to FILEA1, then the program will stay in this waiting state; the CANCEL operator control command must be entered to free the partition.

Considerations and Restrictions

To avoid file contention problems when using doubly-defined files, write the changed records to disk after each update.

When the system is executing a program with a file that can be shared, the parameter SHARE-YES can be included on the FILE statement. SHARE-YES allows file sharing between programs executing at the same time if the access methods used are compatible (see Figure 2-1). SHARE-NO disallows file sharing. If this parameter is not given and RETAIN-S was not specified, SHARE-YES is assumed. (Scratch files are not shared.)

Several restrictions apply to file sharing. For programs specifying double buffering for a file to be shared, the double buffering request is ignored; single buffering is used. Offline multivolume files cannot be shared.

A maximum of four online multivolumes can be shared. Random and sequential access methods are supported under batch; only random access is supported under CCP. The default on multivolume OCL is SHARE-YES.

When a particular file is referenced more than once within a job step, SHARE-NO cannot be specified.

Disk file sharing is also supported by device independent data management disk files.

A display of the system status (via DISPLAY STATUS command) includes the size of the file share area.

For additional information on file sharing, see *Multiprogramming Considerations and Restrictions.*

File Open As \ Attempt To Open As	CO	CA	CU	CG	DO	DU	DG	IO	IR	IRU	IRUA	IRA	IA	IS	ISL	ISU	ISUL	ISA	ISUA	
CO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Consecutive Output
CA	N	N	Y	Y	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	Consecutive Add
CU	N	Y	Y	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Consecutive Update
CG	N	Y	Y	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Consecutive Get
DO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Direct Output
DU	N	Y	Y	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y[1]	Y[1]	Direct Update
DG	N	Y	Y	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y[1]	Y[1]	Direct Get
IO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Indexed Output
IR	N	N	Y	Y	N	Y	Y	N	Y	Y*	Y	Y	Y	Y	Y	Y	Y	Y[1]	Y[1]	Indexed Random Input
IRU	N	N	Y	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y[1]	Y[1]	Indexed Random Input Update
IRUA	N	N	Y	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y[2]	Y[2]	N	N	Indexed Random Input Update and Add
IRA	N	N	Y	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y[2]	Y[2]	N	N	Indexed Random Input Add
IA	N	N	Y	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y[2]	Y[2]	N	N	Indexed Add
IS	N	N	Y	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y[1]	Y[1]	Indexed Sequential Input
ISL	N	N	Y	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y[1]	Y[1]	Indexed Sequential Input with Limits
ISU	N	N	Y	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y[1]	Y[1]	Indexed Sequential Input Update
ISUL	N	N	Y	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y[1]	Y[1]	Indexed Sequential Input Update with Limits
ISA	N	N	Y	Y	N	Y	Y	N	Y[1]	Y[1]	N	N	N	Y[1]	Y[1]	Y	Y	N	N	Indexed Sequential Input Add
ISUA	N	N	Y	Y	N	Y	Y	N	Y[1]	Y[1]	N	N	N	Y[1]	Y[1]	Y	Y	N	N	Indexed Sequential Input Update and Add

[1] Index Sequential Adds may be used one time as the first add access to the file. If any adds have previously been performed and key sort has not occurred, only random access methods may be used to add to the file.
[2] If random add has taken place, the added records are not accessed.

Notes:

1. Above access methods indicate the data management accessing the file, not the file type.
2. Additional information about the availability of added records is included under *Multiprogramming Considerations and Restrictions* in Part 2 of this manual.
3. If you are using the pseudo tape access method (PTAM), no file sharing is allowed between partitions. (PTAM is the access method used for FORTRAN sequential files.)

Figure 2-1. Compatible Disk Access Methods for 5704-SC2 File Sharing

General Results When the 2 or 3 Option is Selected for a Message

For most messages where option 2 or 3 is selected, the job or job step is cancelled with no further action being taken. Exceptions are noted in the message list contained in the *IBM System/3 Model 15 System Messages,* GC21-5076. The reason for the message indicates the corrective action that must be taken before the job or job step can be resubmitted. In some cases, it may be necessary to contact a service representative to correct the problem.

If a 2 or 3 option is taken to a message during a $MAINT run, the library or library entries involved in the function might be destroyed.

Considerations When Selecting the 2 Option

When the 2 option is selected to an OCL diagnostic issued before the LOAD statement is read, the 2 option is treated as a 0 option.

When the 2 option is selected to a system message and the partition is in job mode, all remaining steps in the job are also cancelled. On systems that have input spooling, job processing in the partition will resume with the next job on the input queue. On systems that do not support input spooling, the job stream will be read but not processed until a JOB or /. statement is read. If output spooling is supported on the system, spooled output created by the job will be saved.

When the 2 option is selected to a system message and the partition is in step mode, only the current job step will be cancelled. The job stream will be read but not processed until a JOB, LOAD, CALL, /., or /& statement is read.

When a message occurs during the execution of a job step and the 2 option was selected, the disk files will reflect all activity up to the point when the message occurred and the 2 option was selected. However, if the message occurs during a $MAINT run, the selection of the 2 option will not cause the library or library entries involved in the function to be retained.

Note: The 2 option cannot be taken when messages are issued during RPG II last record (LR) time.

Considerations When Selecting the 3 (or D) Option

When the 3 option is selected to a system message and the partition is in job mode, all remaining steps in the job are cancelled. On systems that have input spooling, job processing in the partition will resume with the next job on the input queue. On systems that do not support input spooling, the job stream will be read but not processed until a JOB or /. statement is read. If output spooling is supported on the system, spooled output created by the job will be saved.

When the 3 option is selected to a system message and the partition is in step mode, only the current job step will be cancelled. The job stream will be read but not processed until a JOB, LOAD, CALL, /., or /& statement is encountered.

When a message occurs during the execution of a job step and the 3 option is selected, the status of the disk files being used by the program depends on the operations being performed. The possible disposition of the disk files are:

- New files being created will not be retained.
- Old files being deleted will be retained.
- Old files being added to will not reflect the additions unless opened as SHARE-YES. If the file is shared, the additions will be reflected in the VTOC.
- Old files being updated will reflect the updates to the point at which the message occurred. (Updated records residing in main storage buffers at this time will not be reflected in the data file.)
- Shared files will reflect all activity up to the point when the message occurred, similar to the 2 option.

WORK FILES

A work file is space required by the language translators (RPG II, COBOL, FORTRAN, CCP/Disk Sort, and Basic Assembler) and several system programs (macro processor and overlay linkage editor). A work file may be used in processing large indexed disk files (see *Large Index Files* in this section). A work file can be specified via a FILE statement or can be obtained by auto-allocate.

The Model 15 language translators require work files for compilation. If an object program is required, work files for the overlay linkage editor are also required. The following estimates of work file space requirements can be used for planning purposes; they may not be valid in all circumstances.

The number of tracks for each work area depends on the area of the disk used. A simulation area has 24 sectors (6,144 bytes) per track; the main data area has 48 sectors (12,228 bytes) per track. Either area can be used for all components.

Main Storage Requirements

Component	Work File Names	Minimum Main Storage Requirements[2]	
		Compile	Execute
RPG II Compiler	$SOURCE $WORK	10K	2K
COBOL Compiler	$SOURCE $WORK $WORKX	12K[1]	8K
FORTRAN Compiler	$SOURCE $WORK	10K	8K
CCP/Disk Sort	$SOURCE $WORK	12K	12K
Basic Assembler	$SOURCE $WORK $WORK2	10K	2K
Macro Processor	$SOURCE	n/a	12K
Overlay Linkage Editor	$SOURCE $WORK	n/a	10K

[1] 14K if braille output is required.
[2] Minimum partition that can be used is 8K.

RPG II

The RPG II compiler (Program Number 5704-RG2) requires two work files. The size of these files depends on the number of source statements (excluding comments) and the type of statements in the program. For the following table, a *compressed* source statement length of 50 bytes is assumed.

Number of Source Statements	Number of Tracks for $SOURCE/$WORK	
	Simulation Area	Main Data Area
50	3	2
100	3	2
150	4	2
200	4	2
300	5	3
400	6	3
500	6	3
700	8	4
1000	11	6
1500	14	7
2000	18	9

To generate an object program, the requirements for the overlay linkage editor must also be met. The size of $SOURCE and $WORK must be large enough to accommodate both the RPG II compiler and the overlay linkage editor.

COBOL

COBOL (Program Number 5704-CB2) requires disk space for each of its three work files. The amount of space required depends on the number of statements in the source program. The following chart shows requirements for each of the three work files.

Number of Source Statements	Number of Tracks for $SOURCE, $WORK, or $WORKX	
	Simulation Area	Main Data Area
50	1	1
100	2	1
150	3	2
200	4	2
300	6	3
400	8	4
500	10	5
1000	20	10
2000	40	20

In addition, to generate an object program, the requirements for the overlay linkage editor must be met.

FORTRAN

FORTRAN (Program Number 5704-FO2) requires disk work space for $WORK during compilation. This space varies according to the size and complexity of the program and, generally, 10 tracks for a simulation area or 5 tracks for a main data area are sufficient.

$SOURCE is not used by the compiler, but it is allocated so that the overlay linkage editor will have the space available. (The requirements for the overlay linkage editor must be met.) If linking is not required, at least 1 track must be specified for $SOURCE.

CCP/Disk Sort

CCP/Disk Sort (Program Number 5704-SM7) requires disk work space for $WORK during program generation. $WORK is used to store the object program, and its size is therefore relative to the size of the object program—not to that of the source program. Each object record requires 64 bytes, and four entries are stored per sector. Because each simulation area track can contain 96 records, 2 tracks are usually sufficient for most sort generations.

$SOURCE is not used by the generator, but it is allocated so that the overlay linkage editor will have the space available. (The $WORK and $SOURCE files must reside on a simulation area for CCP/Disk Sort.)

In all cases, the overlay linkage editor requirements for $SOURCE and $WORK must be met.

Basic Assembler

The Basic Assembler (Program Number 5704-AS2) requires work space for assembly as shown below. (For more information, refer to *System/3 Basic Assembler Reference Manual,* SC21-7509.) *Number of Source Statements* includes source statements generated (expanded) by the macro processor.

Number of Source Statements	Number of Tracks for $SOURCE		Number of Tracks for $WORK2		Number of Tracks for $WORK	
	Simulation Area	Main Data Area	Simulation Area	Main Data Area	Simulation Area	Main Data Area
100	2	1	2	1	(see below)	
200	4	2	4	2		
300	5	3	6	3		
400	7	4	7	4		
500	8	4	9	5		
600	10	5	11	6		
700	11	6	12	6		
800	13	7	14	7		
900	15	8	16	8		
1000	16	8	18	9		

To generate an object module, the requirements for the overlay linkage editor must also be met. If macros are used, the macro processor is executed before the Basic Assembler, but the requirements are still as shown in the preceding estimates.

$WORK is used to store the object program, and its size is therefore relative to the size of the object program—not to that of the source program. Each object record requires 64 bytes, and four entries are stored per sector. Because each simulation area track can contain 96 records and each main data area track can contain 192 records, 2 tracks are usually sufficient for most assemblies.

Overlay Linkage Editor

The overlay linkage editor ($OLINK program) can be invoked automatically by the compilers or by the assembler. It requires work space as shown below. The size of the work areas depends on the amount of storage available (partition size) rather than on the number of statements.

Partition Size	Number of Tracks for $SOURCE/$WORK	
	Simulation Area	Main Data Area
10K	4	2
12K	4	2
16K	5	3
20K	6	3
24K	6	3
28K	7	4
32K	8	4
36K	8	4
40K	9	5
44K	10	5
48K	10	5
>48K	10	5

As an example in the use of this information, assume that a 200-statement RPG II program is to be compiled in a 32K partition. Work files are to be on a simulation area. RPG II requires 4 tracks, and the overlay linkage editor requires 8 tracks. Therefore, 12 tracks should be specified for $SOURCE, and 12 tracks should be specified for $WORK.

Large Index Files

Work files are also used in processing large indexed disk files. With a large indexed file, the amount of time needed to sort the keys at end of job step may be excessive; this amount of time is most significant when adding a large number of records to the file, performing an unordered load, or executing $COPY with REORG-NO and with OMIT or DELETE specified. This sort time can be reduced if a work file is used. The optional work file can either be allocated by the user or be automatically allocated by the system.

The user allocates the work space by supplying a $INDEX45 or a $INDEX40 FILE statement. Or, the system attempts to allocate the work space if either of the following conditions exists:

- Neither a $INDEX45 nor a $INDEX40 FILE statement is supplied.
- The space requested via a $INDEX45 or a $INDEX40 FILE statement is not large enough.

To automatically allocate the work space, the system always checks for sufficient space on the main data area of D1 first; if sufficient space is not available on D1, the system then checks the main data area on D2; if the space on D2 is insufficient, the search continues on the available main data areas until either sufficient space is found or a message is issued to indicate that the system cannot automatically allocate the work space.

The restriction on the maximum number of files allowed by the system applies to all files automatically allocated by the system as well as the files allocated by the user. Therefore, in order to use the work file, you must not have previously specified the maximum number of files allowed.

If you wish to provide the work space with a $INDEX45 or a $INDEX40 FILE statement, the following information will aid you in determining the required size.

The work file, called $INDEX 45 or $INDEX40 is used to sort the added keys and then merge the added keys into the index and must be large enough to contain all of the keys added to the file. If the program adds records to more than one indexed file, the work file must be large enough to contain all the keys for the file whose added keys occupy the greatest key index area. The work file should be as close as possible to the beginning of the file whose keys are being sorted or on a different disk drive. This arrangement minimizes the disk seek time.

The work file must be named $INDEX45 or $INDEX40 and be located on a main data area. To determine the number of tracks required for the work file, use the following formula:

$$\text{Number of adds} \div \frac{256}{(\text{key length} + 4)} \div 48 = \text{Tracks for main data area}$$

After dividing 256 by key length + 4, the remainder should be dropped. After the other divisions, round the quotient to the next highest whole number.

The work file can be used with multivolume files. However, it cannot be located on an area that contains one of the offline volumes of a multivolume file. The data module containing the work file must remain online while the program is run. The work file must be RETAIN-S. If RETAIN-T or RETAIN-P is specified, the system forces it to RETAIN-S.

For small indexed files of 10 tracks or less where the sort time is negligible, a work file does not improve performance and should not be used.

For this performance option, no change to the source program is needed. Also, programs need not be recompiled to use this option; only one additional FILE OCL statement is needed.

The system allocates disk space for the $INDEX45 or $INDEX40 work file prior to allocating space for any other file(s) used by the program. Therefore, if you specify a location for any file(s) used by the program, you should also specify a location for the $INDEX45 or $INDEX40 work file. This procedure prevents the system from attempting to assign the same disk space to two different files.

For additional considerations on when to specify $INDEX45 or $INDEX40, see *File Share* under *Multiprogramming Considerations and Restrictions.*

MULTIVOLUME DISK FILES

When a file is too large for one main data area, it can be continued on one or more subsequent main data areas; such files are called multivolume files. (A volume is one main data area.) Multivolume files are not supported on simulation areas. Multivolume files can be online or offline. A file is online if all volumes are mounted when the program begins. An online file has an equal number of UNIT and PACK parameters; an offline file has fewer UNIT parameters (shares same unit). Offline multivolume files can only be used on drives 1 (when an IPL is performed from 3344 drive 3) and 2.

The way in which a disk multivolume file is created depends on the file type. For a sequential or indexed file, the records are stored in consecutive locations, in the order they are read. One main data area is filled at a time. For sequential files, each volume must be filled before the next volume is allocated and loaded. For indexed files, each volume need not be filled. Each indexed volume is loaded until a key field is reached that is higher than the HIKEY for that volume; then the next volume is allocated and loaded. Indexed files must be loaded in key field sequence. A message occurs if a volume is filled and there is no record with a key field equal to the HIKEY for that volume.

For example, suppose the HIKEY for a volume is 166, and a record with the key field 162 is loaded. It is less than the HIKEY, so it is loaded on the volume. Next, a record with the key field 170 is loaded. Record 170 is loaded on the next volume, and an error message occurs. The reason for the error message is that a key field record equal to 166 was not loaded before loading records to a new volume. This error message can be bypassed. A record can be loaded on the next volume, and at some future time a key field record less than or equal to the HIKEY can be inserted. A random add is required to add a record higher than the highest key on the volume but lower than or equal to the HIKEY.

Indexed and sequential files may be either online or offline.

Removable data modules can be used when sequential or indexed files are created. A data module is mounted, the system indicates when it is full or the high key is reached, and then the next data module is mounted. When two drives are available, two data modules can be mounted. When the first one is *completed*, it can be replaced with a third while the program *processes* the second data module. In either case, no more than 192 volumes (96 for indexed files) can be used per program.

Space can be allocated on all volumes of a multivolume file if the volumes are online at the time of the allocation. Space can also be allocated for an offline file, other than the initial volume, but the volumes must be empty or space known to be available. Both fixed and removable data modules can be used with any online multivolume file. Space for a volume of a multivolume file is reserved after one or more records are placed in that volume.

Direct files must be online. The maximum number of volumes, therefore, is two on a two-drive 3340 system, or four on a four-drive 3340 system.

Processing offline multivolume files depends on the access method a program uses. If records are read from a sequential or indexed file, a data module is mounted; when all the records have been read from the data module, the next data module is mounted. With two drives, two data modules are mounted; when all the records have been read from the first data module, that data module may be replaced with the third while the program reads from the second data module. When online, any combination of fixed and removable data modules is acceptable, but all must be mounted and must remain mounted. Only four volumes will be used during online random processing (direct or indexed random accesses) regardless of the number specified in the OCL.

MULTIVOLUME TAPE FILES

A tape file may also be too large for one tape and can be continued on one or more tapes. Such files are called multivolume tape files. (A volume is one tape.)

When end of volume is reached on a multivolume tape file, that volume rewinds to load point and unloads. If the drive that is to contain the next volume (whether the same drive or another drive) is not in a ready condition, an error message is issued.

Processing continues when the drive that is to contain the next volume is made ready. When using alternating drives, if the next volume is mounted and the drive is ready when end of volume is reached, processing continues without stopping.

MULTIFILE TAPE VOLUMES

Just as a disk normally contains more than one file, more than one file can exist on a reel of tape. If this is the case, each file has an associated file number. The system uses the SEQNUM parameter on the tape FILE OCL statement to indicate the file number and position the tape to the desired file.

The sequence number indicates the relative position of the file on the tape volume and is incremented by one from one file to the next.

Null Files on Tape

A null file is a file on disk that has no data records in it; for example, a null file may occur if a user's error or exception file has no records on some days but does have some records on other days. If the user has a procedure that always backs up that file to tape, it is important to know what happens to null files on a multifile tape volume.

Unlabeled Tapes

A multifile tape volume can contain all unlabeled files. When a file is copied to tape, it is followed by two tape marks (TM):

```
FILE1 TM TM
```

If a second file is added to this tape and a SEQNUM-2 parameter is specified, and if that file contains data, it will result in:

```
FILE1 TM FILE2 TM TM
```

But if the second file (FILE2) contains no data, the result is:

```
FILE1 TM TM TM
```

And if a third file (FILE3) is added to this tape, SEQNUM-2 must be specified and the result is:

```
FILE1 TM FILE3 TM TM
```

In this example, if the third file is added, and SEQNUM-3 is specified instead of SEQNUM-2, a halt will be issued unless END-LEAVE had been specified for the second file.

In any event, with unlabeled tapes, a null file is not saved.

Labeled Tapes

With labeled tapes, however, null files can be saved on tape and later restored to disk. If one file containing data is written to tape, it looks like this:

VOL HDR TM FILE1 TM TL TM TM

where VOL represents the volume label, HDR represents the header labels, TL represents the trailer labels, and TM represents a tape mark.

If a second file is added to this tape and a SEQNUM-2 parameter is specified, and if that file contains data, it will result in:

VOL HDR TM FILE1 TM TL TM HDR TM FILE2 TM TL TM TM

But if the second file contains no data, the result instead is:

VOL HDR TM FILE1 TM TL TM HDR TM TM TL TM TM

And if a third file is added to this tape (SEQNUM-3), the result is:

VOL HDR TM FILE1 TM TL TM HDR TM TM TL TM HDR TM FILE3 TM TL TM TM

If the third file (FILE3) is added and SEQNUM-2 is specified instead of SEQNUM-3, then the result is as follows:

VOL HDR TM FILE1 TM TL TM HDR TM FILE3 TM TL TM TM

Note that in any case if a file is added to a multifile tape, it is assumed to be the last file on the tape and all subsequent files on the tape are not accessible.

PROGRAMMING CONSIDERATIONS

If the root segment of a program is too large, the system may issue an error message while attempting to open a disk file. This message indicates that certain fields are located at a main storage address above logical 40K (in a batch partition) or above 24K (for a CCP task). Normally, this message occurs only if the program has large I/O areas. Following are some actions that may be tried in attempting to circumvent this message:

- Specify SHARE-NO for files used in a batch partition.
- Reduce the block size of files in the program.
- Remove double buffer specifications.
- Reduce the size of the core index.
- Recompile the program specifying a smaller execution size (for example, 40K for a batch program; 24K for a CCP program).

For additional information, see *Disk Device Support* in the *System/3 Model 15 System Control Programming Macros Reference Manual,* GC21-7608.

Using Program Number 5704-SC2, any open file is closed when end of job occurs or a 2 or 3 option is taken to a message. Using Program Number 5704-SC1, open files are closed at end of job or when a 2 option is taken to a message; open files are not closed when a 3 option is taken to a message.

COBOL, FORTRAN, and assembler users can control file open and close; RPG II users cannot control open and close. As a result, each program or subprogram must explicitly close any file that is opened.

Library Facilities

LIBRARY DEFINITION

A library is a space on disk used for storing programs and procedures. There are two types of libraries: source and object. Source libraries contain source programs and procedures of OCL statements; object libraries contain executable object programs (load modules) and nonexecutable object programs (routines). Libraries are only located on simulation areas, not on main data areas.

The System/3 library maintenance program is used to:

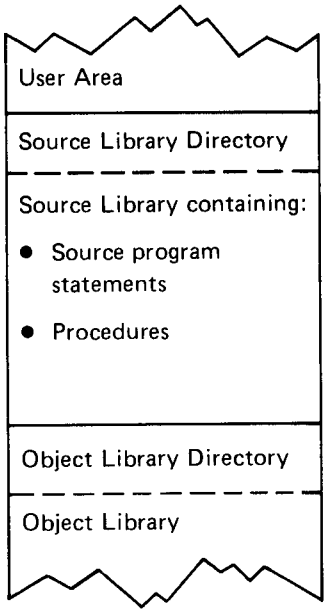
- Create libraries
- Enter source programs, OCL statements, and object programs into libraries
- Maintain libraries
- Create files containing library entries

The library maintenance program creates a separate directory for each library. Every library entry has a corresponding entry in its library directory. The directory entry contains such information as the name and location of the library entry. The program also creates a system directory that contains information about the size and available space in libraries and their directories.

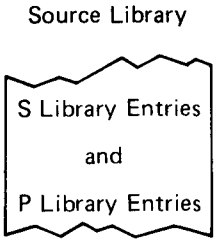
SOURCE LIBRARY

A source library is a disk space for storing source programs, specifications, and procedures. Source programs are sets of user instructions, the most common of which are RPG II source programs and disk sort sequence specifications. Procedures are groups of OCL statements used to load programs and may be followed by input data for the programs. (Procedures for system service programs can, for example, contain control statements.)

The following illustration shows the relative location of a source library with respect to the user area and object library:



Source statements and procedures are two logically different types of entries. When these entries are copied into source libraries, they are given different source library designations. Source programs are given an S library designation; procedures are given a P library designation. The following illustration shows the logical entries within the source library:



The S library entries are source programs. Procedures *cannot* be executed from this library.

The P library entries are procedures which can be executed.

This page intentionally left blank.

Physical Characteristics

A source library has the following physical characteristics:

- Size: The minimum size of a source library is one track.
- Directory: The directory acts as a table of contents and contains the name and location of each source library entry. The first two sectors of the first track are always assigned to the directory, with additional sectors used as needed.
- Organization of Entries: Entries (source statements and procedures) within the source library need not be stored in consecutive sectors. An entry can be stored in widely separated sectors. Within each sector is a pointer to the sector that contains the next part of the entry.

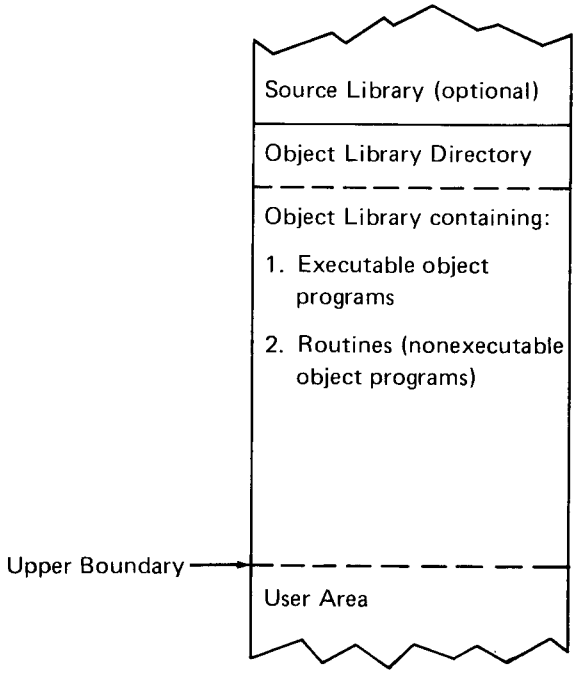
  The boundary of the source library cannot be expanded; therefore, an entry must fit within the available library space. The system provides maximum space within the prescribed limits of the source library by compressing entries. That is, all duplicate characters are removed from the entries. Later, if the entries are printed or punched, written to diskette, or copied to a file, the duplicate characters are reinserted.
- Location: A source library can only be located on a simulation area, not on a main data area. There is only one source library per simulation area; however, source libraries may reside on several simulation areas.

  Note: When the size of the source library is changed or the source library is reorganized, all temporary entries are deleted.

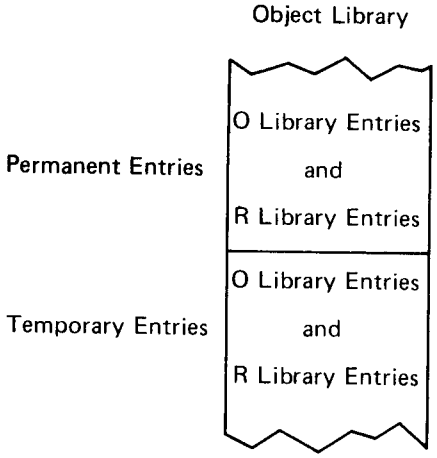
OBJECT LIBRARY

An object library is a disk area used for storing object programs and routines. Object programs (also known as load modules or executable programs) are programs and subroutines that can be loaded for execution. Routines (or nonexecutable programs) are programs and subroutines that need to be link-edited before they can be loaded for execution. Routines are used by compilers and (except for FORTRAN) must be on the same simulation area as the compiler using it.

The following illustration shows the relative location of the object library with respect to the source library and user area:



The object library contains two logically different types of entries: object programs and routines. When these entries are copied into the object library, they are given different object library designations. Object programs are given an O library designation; routines are given an R library designation. The following illustration shows the logical library entries within the object library.



The *O* library entries are executable programs. They are loaded by the LOAD statement.

The *R* library entries are nonexecutable routines used by the compiler.

Physical Characteristics

An object library has the following physical characteristics:

- Size: The size of the object library depends on whether or not the library is on a system pack (a simulation area containing the system programs). An object library can be created on any simulation area, but one library containing the system programs must be online. The minimum size of an object library is 3 tracks.

  The disk area for an object library consisting of system programs must also be large enough to contain a scheduler work area for disk system management and a system history area. The number of tracks for the scheduler work area and the system history area are not included in the number of tracks specified for the library; the library maintenance program calculates and assigns the additional space. (See *Scheduler Work Area* under *File Facilities*.)

- Directory: The directory acts as a table of contents and contains the name and location of the object library entries. If the object library is on a system pack, three of the requested tracks are reserved for the directory. If not, only the first track is reserved for the directory. The user may override these directory sizes by specifying the DIRSIZE parameter on the ALLOCATE statement of the library maintenance program.

- Upper Boundary: When copying temporary entries into the object library, the upper boundary of the library automatically expands as additional tracks are needed if the space following the object library is available. The upper boundary of the library is extended to the end of the simulation area or to the first temporary or permanent file. At the successful completion of the copy, the upper boundary is returned to the track boundary at the end of the last temporary entry.

  If the copy was not completed successfully, the upper boundary may remain extended. When a permanent entry is placed in the library or the library is reorganized, all temporary entries are deleted and the upper boundary returns to its original location. Permanent entries cannot exceed the original upper boundary.

  To make efficient use of this feature, the area next to the upper boundary of the object library should be kept free of data files. When disk system management automatically allocates file space, the area next to the object library is probably free because the files are placed as close to the high end of the simulation area as possible. Users should also allocate files toward the high end of the simulation area. This leaves room for object library expansion.

- Organization of Entries: Entries are stored in the object library serially; that is, a 20-sector program occupies 20 consecutive sectors. Temporary entries follow all permanent entries in the object library.

Maintaining an Object Library

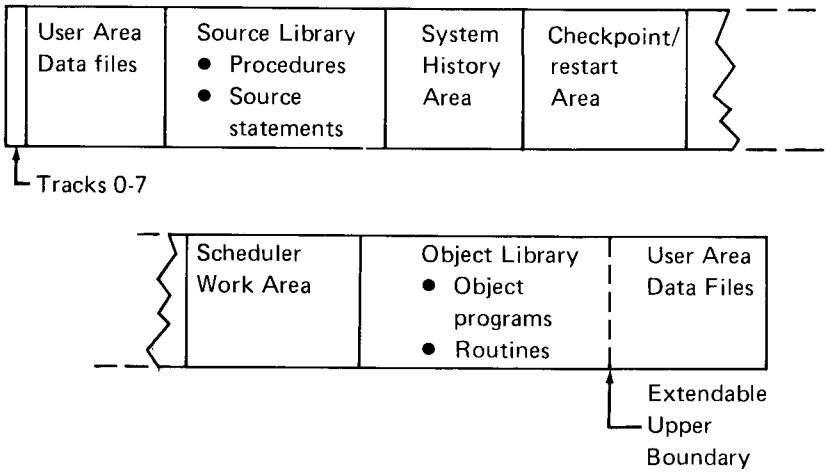
Gaps can occur in the object library when entries are deleted. The associated directory entries point to these gaps. When the library maintenance program places a new permanent entry in the library, it searches the directory for a gap that has the same number of sectors, or the fewest sectors over the number required by the new entry. If the entry is smaller than the gap, the last part of the gap is not pointed to by a directory entry. Since this gap has no directory entry, it cannot be used until the library is reorganized.

The library maintenance program should be used to reorganize the library when a great number of additions and deletions have been done (creating an excessive number of unavailable sectors), or when there is no apparent room. In reorganizing entries, the library maintenance program deletes temporary entries and shifts entries so that gaps do not appear between them. This reorganization makes more sectors available for use.

By printing the system directory, the library maintenance program can be used to determine how many sectors are available.

LIBRARY LOCATIONS

Libraries can be located by the system anywhere on a simulation area. However, the location of a source library with respect to an object library is always the same:



If space is allocated for only a source or object library, the library maintenance program places the library in the first available disk area large enough to contain the library. When allocating space for a source library on a simulation area containing an object library, an area large enough for the source library must immediately follow the object library.

The library maintenance program moves the object library to allow space for the source library to precede it. If an object library is being allocated on a simulation area with a source library, space for the object library must immediately follow the source library.

STORING PROGRAMS

Three methods are available for storing programs into libraries: the library maintenance program, a specification on the RPG II Control Card, and the COMPILE OCL statement.

Library Maintenance Program: Depending on user specifications, the library maintenance program can store programs by copying entries from one location to another (giving new names) within a library; copying entries from one library to another giving new names if required; copying entries from the system input device to a library; or copying entries from a file to a library.

(See Part 4 for additional information on the library maintenance program.)

RPG II Control Card: RPG II can indicate the type of object program output after the system compiles a source program. The compiled program can be stored in an object library, punched into cards, or written to a diskette. A program written as a temporary entry in the object library is deleted by the next program written permanently in the object library or by the next program of the same name written as a temporary entry in the object library. The object program is written in the object library that contains the compiler, unless a COMPILE statement indicates otherwise.

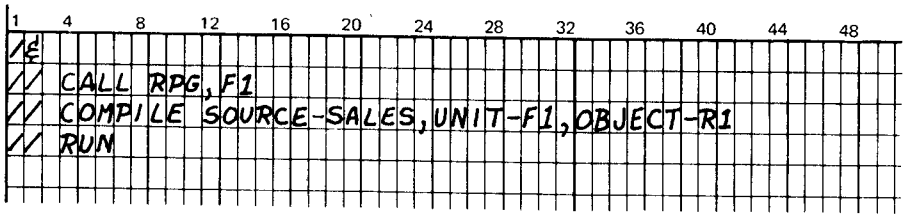
Column 10 on the RPG II Control Card is used to specify the object output. Columns 75–80 are used to name the object program. When a name is not assigned, RPGOBJ is assumed. For detailed information on these specifications, see the *IBM System/3 RPG II Reference Manual,* SC21-7504.

COMPILE OCL Statement: The COMPILE OCL statement tells the system to:

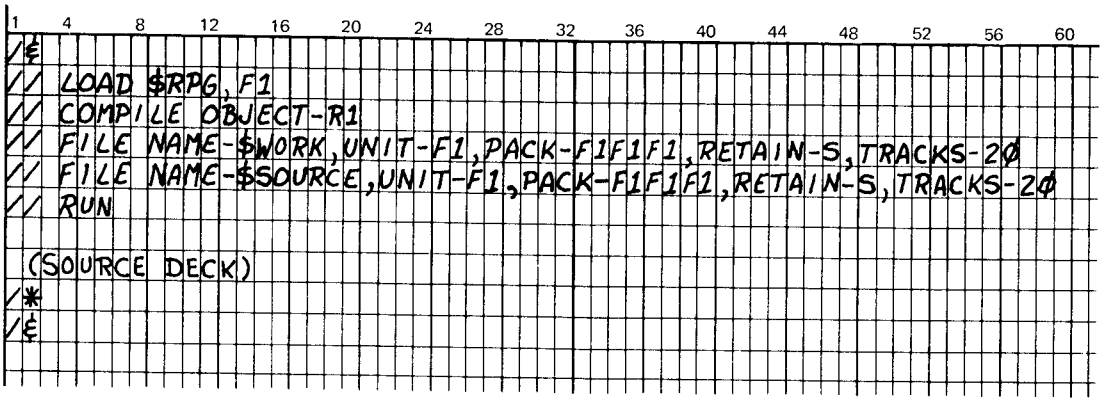
- Compile a source program from a source library and store the object program in an object library, or
- Compile a source program from the system input device and store the object program in an object library

For the format of this statement, see *COMPILE statement* in Part 1 of this manual.

Sample Statements



This sample job stream tells the system that the source program named SALES is located on F1. The OBJECT-R1 keyword parameter tells the system to place the object program on R1.



This sample job stream compiles a source program from the system input device and stores it in an object library on R1. If the OBJECT parameter was not coded, the program would be compiled and placed into the same object library as the compiler (F1).

PROCEDURES

Procedures are sets of OCL statements in a source library. Procedures are stored in the source library via the library maintenance program. Since the records in the source library are compressed, a nested procedure containing a CALL statement should not be modified by a program run as part of that nested procedure. A procedure cannot contain more than one LOAD statement and cannot contain any JOB statements. All other OCL statements, except /&, /*, and /., are allowed in procedures. The CALL statement is allowed only in nested procedures. LOAD * statements are allowed in procedures; however, the object program must be read from the system input device.

A maximum of 31 control statements can be included in procedures for the system service programs. The control statements must follow the OCL statements in the procedure. A RUN statement must be the last OCL statement in the procedure to separate the OCL statements from the control statements. The RUN statement in the job stream, rather than the one in the procedure, causes the system to run the program.

The following example shows how to create a procedure with the library maintenance program ($MAINT). The procedure name is PROC1 (NAME-PROC1 in COPY statement). This name identifies the procedure in the source library. The procedure is placed in the source library on F1 (TO-F1 in COPY statement). This procedure is referred to in all of the following examples.

OCL being copied to the source library as a procedure

```
// LOAD $MAINT,F1
// RUN
// COPY FROM-READER,TO-F1,LIBRARY-P,NAME-PROC1,RETAIN-P
// LOAD ENDMON,R2
// FILE NAME-DALTOT,UNIT-F2,PACK-VOL04,RECORDS-1500,RETAIN-P
// FILE NAME-ACCTOT,LABEL-TOTAL,UNIT-R1,PACK-VOL02,DATE-12/4/76
// SWITCH XXX01XX0
// RUN
// CEND
// END
```

To merge the procedure (unchanged) into the job stream, two statements are used in the job stream; this is the normal procedure call.

```
// CALL PROC1,F1
// RUN
```

Parameters in any of the statements (except the INCLUDE statement) in a procedure for the first job step can be changed by procedure override statements placed between the CALL and RUN statements. Procedure override statements modify the procedure for the first job step. For example, the following changes can be made to procedure PROC1:

- In the first FILE statement (NAME-DALTOT), change the RECORDS parameter from RECORDS-1500 to RECORDS-1750.
- Change the parameter in the SWITCH statement from XXX01XX0 to XXX10XX1.

The following statements are needed in the job stream to call and modify PROC1. Note that the NAME parameter is also supplied in the FILE statement. This is necessary to identify the FILE statement to which the change applies. Therefore, the NAME parameter cannot be changed.

```
1   4   8   12  16  20  24  28  32  36
//  CALL PROC1,F1
//  FILE NAME-DALTOT,RECORDS-1750
// SWITCH XXX10XX1
//  RUN
```

Note: If a valid override statement is used to change a statement which contains an OCL error, an OCL error message will still be issued.

Besides changing a parameter, a parameter in a procedure statement can be entirely deleted if it is a keyword parameter. To delete a parameter in any of the statements, the keyword and hyphen are coded followed immediately with a comma. A PACK or REEL parameter (which is to be deleted by a procedure override statement) must precede the parameter that is to replace it. The procedure override statement in the following example deletes the RETAIN parameter completely.

```
1   4   8   12  16  20  24  28  32  36
// CALL PROC1,F1
// FILE RETAIN-,NAME-DALTOT
// RUN
```

Statements can be added to a procedure by placing the added statements between the CALL and RUN statements. For example, to add a NOHALT statement to the procedure, the following statements are needed in the job stream:

```
1   4   8   12  16  20  24  28  32  36
// CALL PROC1,F1
// NOHALT
// RUN
```

Parameters can be omitted from all OCL statements in a procedure and then supplied between the CALL and RUN statements. For example, assume the procedure contained the LOAD statement shown.

```
1   4   8   12  16  20  24  28  32  36
//  LOAD ,R2
```

To run the ENDMON program, the entire LOAD statement did not have to be supplied; only the missing parameter was included in the job stream.

```
1   4   8   12  16  20  24  28  32  36
//  CALL PROC1,F1
//  LOAD ENDMON
//  RUN
```

Example

Procedure override statements are logged on the system log device along with the statements in the job stream.

```
1   4   8   12  16  20  24  28  32  36
// CALL PROC1,F1
// FILE NAME-DALTOT,RECORDS-1750
// SWITCH XXX10XX1
// NOHALT
// RUN
```

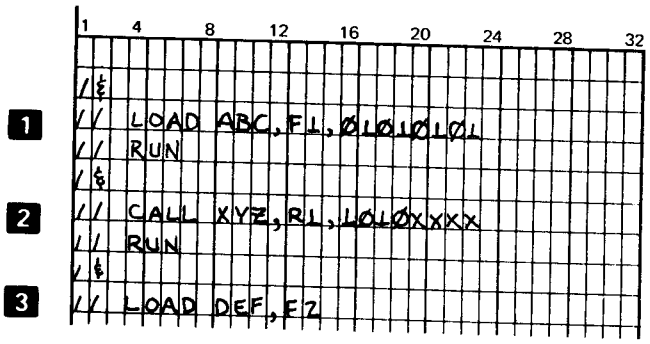
The statements from the procedure are merged with the preceding statements and printed.

```
// CALL PROC1,F1
XX LOAD ENDMON,R2
XX FILE NAME-DALTOT,UNIT-F2,PACK-VOL04,RECORDS-1500,RETAIN-P
// FILE NAME-DALTOT,RECORDS-1750
XX FILE NAME-ACCTOT,LABEL-TOTAL,UNIT-R1,PACK-VOL02,DATE-12/4/76
XX SWITCH XXX01XX0
// SWITCH XXX10XX1
XX RUN
// NOHALT
// RUN
```

Statements preceded by XX represent the procedure statements as they appear in the source library. The CALL and RUN statements and any statements intended as overrides to procedure statements or additions to the procedure begin with //.

Switch characters can be used to determine whether or not a job step will be executed. The following example shows how they are used to call a procedure based on the results of a previous program.

Assume that a program has just completed executing successfully and has set the external indicators to 10101010. The OCL statements following the completed program are:



1 Step 1 is not executed because the external indicators do not agree with the switch characters. An informational message with the value of the external indicators is issued. The system then flushes to the next step.

2 Step 2 is executed because the external indicators agree with the switch characters. (The switch character X tells the system to cancel the compare operation with the relative external indicator.)

3 Step 3 is executed after step 2 is completed.

Nested Procedures

Some procedures are done in the same order every time a job is performed. Nesting procedures is a convenient way to link the procedures and requires a call to only the first procedure. Each procedure calls the next procedure until the job has been completed.

Nested procedures provide several benefits:

- Programs are always run in the correct sequence.
- Operator intervention (and chance of operator error) is decreased.
- Files are less likely to be destroyed as a result of running nonrelated programs between steps of a job.

Here is an example of how nested procedures might be used to back up a main data area containing files that will be used in the future. The OCL statements and control statements to be entered from the system input device to copy one main data area (D3) to another main data area (D2) would look like this if nested procedures were not used:

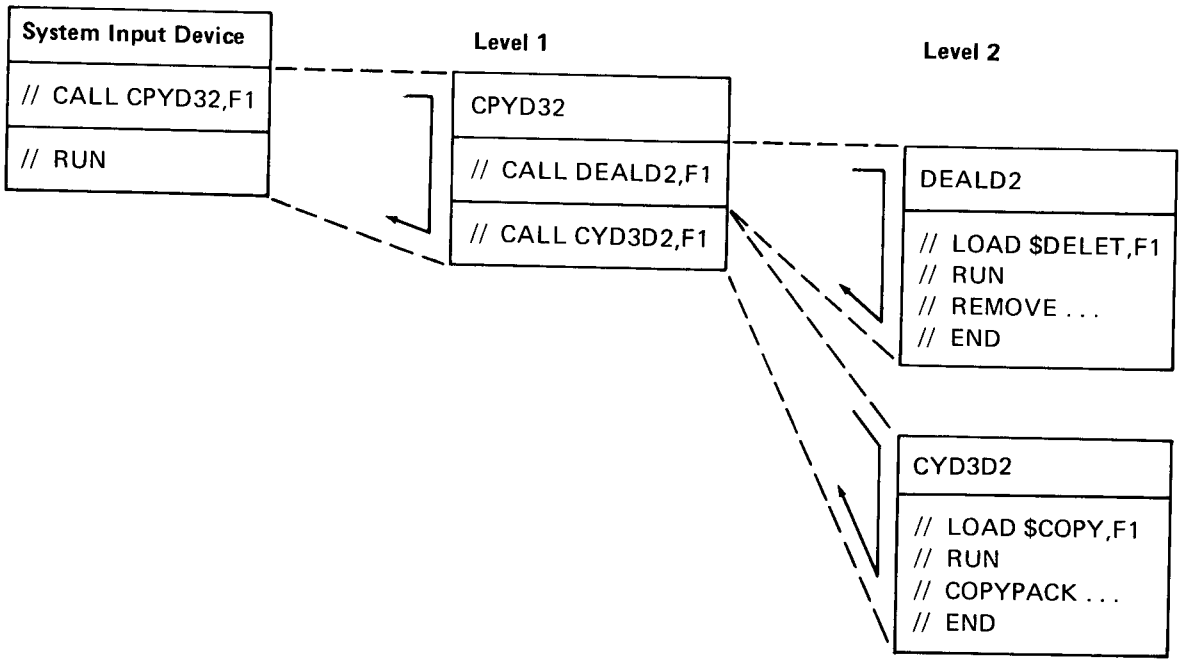
```
1   4   8   12  16  20  24  28  32  36  40  44  48
// LOAD $DELET,F1
// RUN
// REMOVE UNIT-D2,PACK-XXXXXX,LABEL-VTOC
// END

// LOAD $COPY,F1
// RUN
// COPYPACK FROM-D3,TO-D2
```

Assume that the preceding OCL statements were placed in the source library as procedures with the names DEALD2 and CYD3D2, respectively. The use of nested procedures allows building a procedure to call other procedures. The following example shows how a nested procedure is built to call the procedures DEALD2 and CYD3D2:

Placed in the source library as a procedure named CPYD32

```
1   4   8   12  16  20  24  28  32  36  40  44  48  52  56
// LOAD $MAINT,F1
// RUN
// COPY FROM-READER,TO-F1,LIBRARY-P,NAME-CPYD32,RETAIN-P
// CALL DEALD2,F1
// CALL CYD3D2,F1
// CEND
// END
```

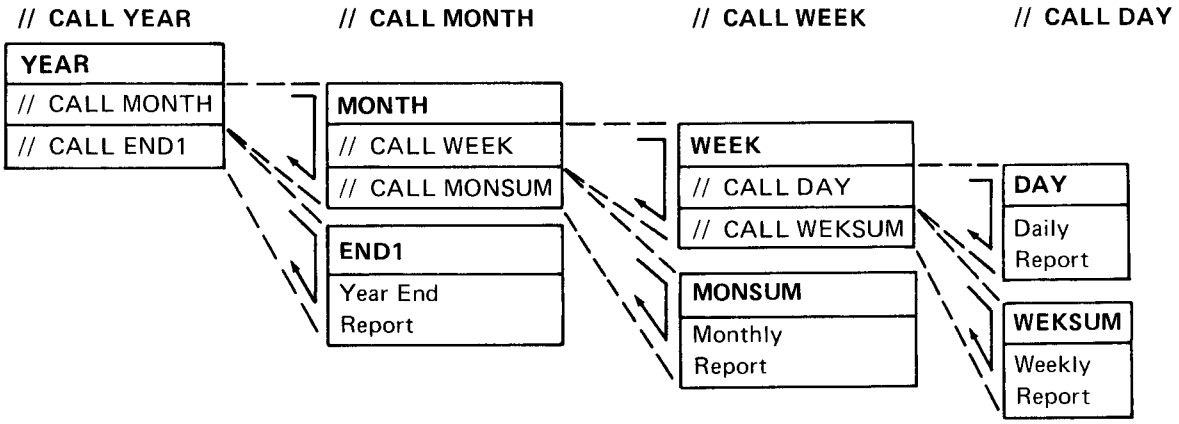
To call the two procedures needed to perform the copy job described, only one CALL statement is required in the job stream from the system input device.



This CALL statement links the job stream to the nested procedure (CPYD32) used to call the procedures necessary to perform the copy job. CPYD32 contains two CALL statements that call the two procedures necessary to copy D3 to D2.

Notice that CPYD32 contains only CALL statements. Any procedure within nested procedures can consist entirely of CALL statements and does not need a RUN statement to indicate the end of the procedure. Nested procedures allow an unrestricted number of CALL statements in a procedure. Therefore, CPYD32 could have more than two CALL statements if it were necessary to add any procedures.

For example, a company issues daily reports on goods bought and sold by calling the DAY procedure. By nesting procedures together, // CALL WEEK can be used to write a daily report and a weekly report. Once a month // CALL MONTH is used to write out daily, weekly, and monthly reports. Finally, daily, weekly, monthly, and yearly reports are written once a year by the YEAR procedure, which nests all of the other procedures together.



No more than nine levels of CALL procedures can be nested together. Levels of procedures are determined by the number of CALL statements away from the system input device a procedure is located. In this example, when // CALL YEAR is given in the system input device, the YEAR procedure would be one level away from the system input device. MONTH and END1 procedures are two levels away from the system input device when // CALL YEAR is given.

With nested procedures, fewer OCL statements and control statements for system service programs are needed in the job stream from the system input device. However, certain rules must be followed to make nested procedures work:

- No more than nine levels of procedures are permitted.
- Each procedure may have an unrestricted number of CALL statements to the next level of procedures.
- Only control statements for system service programs and specifications for disk sort can follow a RUN statement; 31 control statements can be included in a procedure.
- Procedure additions or overrides supplied between the CALL and RUN statements in the job stream are merged only between the first LOAD and RUN statements encountered in the procedures.
- Procedure overrides are merged with the first similar (same identifier) statement before the RUN of the first LOAD and RUN encountered in the procedures, except for FILE statements, which are merged based on the NAME parameter.
- Any OCL statements permitted before the RUN statement in the job stream are also permitted anywhere before the RUN statement in a procedure.

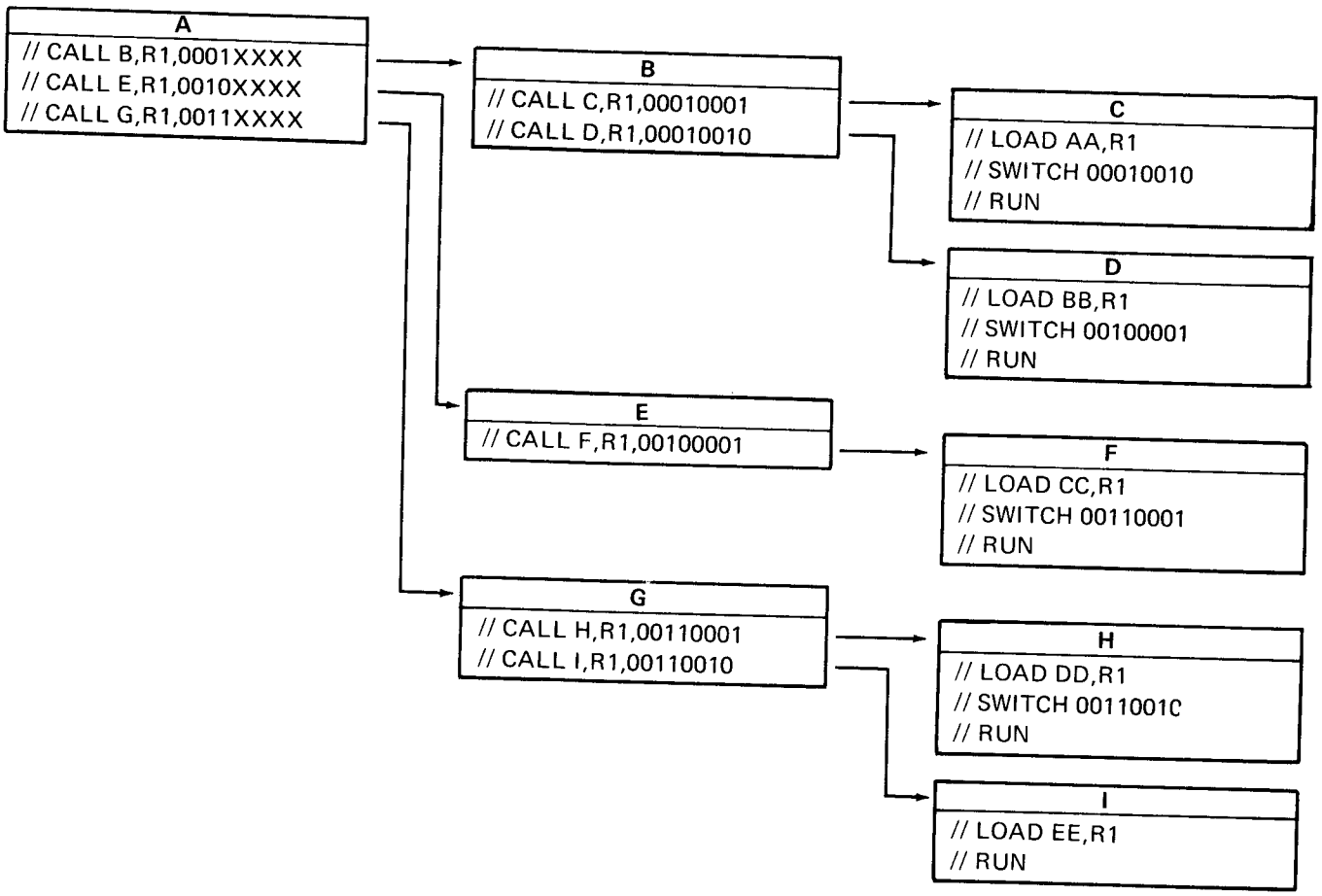
Example of Nesting Procedures

As an example of nesting procedures, assume the NOHALT statement is to be used to decrease operator intervention. In the illustration of nested procedures, the NOHALT statement could be placed between the CALL and RUN statements in the system input device. In this case, it would be read as an additional OCL statement for the DEALD2 procedure. However, it could be placed anywhere in the master procedure, CPYD32, or anywhere before the RUN statement in the DEALD2 or CYD3D2 procedures. The rule would still be followed no matter what procedure contained the additional OCL statement.

Example of Nested Procedure Restart

Changing the value of SWITCH statement (A) permits the following procedure to be restarted from any of the steps within the nested procedure. In this example, the procedure will begin executing at step F because the SWITCH statement sets the external indicators to agree with the switch characters on the // CALL E and // CALL F statements.

(A) // SWITCH 00100001
// CALL A,R1
// RUN



CATALOGING TO AN ACTIVE LIBRARY

Program Number 5704-SC2 enables entries (permanent or temporary) to be stored (cataloged) in an object library located on an active program pack (a simulation area containing a program currently loaded for execution in another partition).

An object program can be cataloged to an active library, but in some cases, a program executing in another partition may be adversely affected. An *active* library is one from which a program is loaded for execution. That library becomes *inactive* after end-of-job step for that program. (Note, however, that a CCP library is always considered *active*.)

If an object program does not have overlays, once it is loaded into a partition for execution, there is no further need to access that library. However, if an object program requires subsequent library accesses (after the program has been loaded for execution), then an access to that library from another partition may cause problems.

Figure 2-2 shows what can happen to another partition when cataloging to an active library.

Some examples of a program requiring subsequent accesses to a library are:

- A program which requires an overlay from disk during execution, for example, an RPG II object program.
- A program which requires phase overlays during execution, for example, a system service program.
- A FORTRAN program which invokes another program from that library.

If all object library entries are permanent entries, then the only conflict arises when cataloging an entry of the same name as the program executing in another partition. (For either permanent or temporary entries, the overlay linkage editor does not allow cataloging a module with the same name as an active program running in a batch partition.)

Cataloging to an active library is an option chosen during system generation. The configuration record program may also be used to change the support selected for cataloging to an active library (see *Configuration Record Program—$CNFIG* in Part 4 of this manual). The following three levels of catalog support are available:

- Disallow catalog to all active libraries. This support is identical to the 5704-SC1 support for cataloging to a program pack. With this level of support, any attempt to catalog an object library entry to an active program pack results in an F/ message. (The overlay linkage editor will issue a message only if the program executing is a LOAD * or a temporary entry.)
- Allow catalog to active CCP libraries but not to other active libraries. This support is identical to 5704-SC1 support for all active libraries except the CCP libraries. No message will be issued when attempting to catalog an object library entry to an active CCP library.
- Allow catalog to all active libraries. No message will be issued when attempting to catalog an object library entry to any program pack.

Cataloging to an active library is supported by both the overlay linkage editor and the library maintenance program.

Activity in Another Partition	Activity in this partition: Cataloging to an active library									
	Temporary Entry					Permanent Entry				
Has a program been loaded[1] into another partition from the active library?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Does the program being cataloged have the same name?	No	No	No	Yes	Yes	No	No	No	Yes	Yes
Does the program in the other partition require subsequent accesses to that library?	No	Yes	Yes	No	Yes	No	Yes	Yes	No	Yes
Is the program executing in the other partition a LOAD[1] program?		Yes					Yes			
Is the program executing in the other partition a temporary program?			Yes					Yes		
Is the other partition affected?	No	Yes	No	Yes	Yes	No	Yes	Yes	Yes	Yes
How is the other partition affected?		Overlay segments could be changed unexpectedly.		The old version of the program, rather than the new version, is executed.	Overlay segments could be changed unexpectedly.		Overlay segments could be changed unexpectedly.		The old version of the program, rather than the new version, is executed.	Overlay segments could be changed unexpectedly.

[1] All of the OCL for this program was processed and the program was fetched into storage for execution.

Figure 2-2. Cataloging to an Active Library

The following example shows the use of the option to catalog to the active CCP libraries:

A user program updating a parts master file is initiated in partition 1 under control of CCP. This program was loaded from the CCP library on F2. While the parts master file is being updated, RPG II (in partition 2) compiles a program to order replacement parts; the object program is cataloged by the overlay linkage editor to the CCP library on F2. (F2 for partitions 1 and 2 is assigned to the same simulation area.) After the parts master file has been updated, the program to order replacement parts can now be executed in partition 1 under control of CCP.

Without the capability to catalog to an active CCP library, it would be necessary to catalog the object program to order replacement parts to a non-CCP library, shut down CCP, copy the object program from the non-CCP library to the CCP library, and start up CCP again before executing the program.

Following is another example showing the use of the option to catalog to the active CCP libraries:

CCP is executing in partition 1, batch programs are executing in partition 2, and programs are being compiled in partition 3. All three partitions use the same program pack. While the batch partition (P2) is executing, programs being compiled (in P3) can be stored in the object library on the simulation area in use as the program pack for the three partitions.

Use of the option to catalog to all active libraries is illustrated by the following example:

A user program performing an inventory analysis was loaded from R1 and is currently executing in partition 2. While the program is printing the report, the library maintenance program, executing in partition 3, can be used to copy the program that calculates a monthly summary analysis into the object library on F2. R1 for partition 2 and F2 for partition 3 are assigned to the same simulation area.

User Considerations

- If you catalog a CCP module for which the program find or format find option was not selected at CCP startup, the old version of the module (or a different module than the one intended) may be used if the new version is placed in a different location in the library.
- If you need to catalog an object module for which a conflict exists (see Figure 2-2), change the other partition to HALT mode, and then perform the catalog operation when the other partition comes to end-of-job step. After the catalog operation, change back to NOHALT mode and continue.

INITIAL PROGRAM LOAD (IPL)

System operation begins with an initial program load (IPL) which clears main storage and transfers the supervisor (a component of the system control program) from disk into main storage.

IPL should be performed every time the system power is turned on, the system pack is changed, or a recovery procedure requires it. It is initiated from one of the following five sources: a card reader (alternate), the F1 simulation area on drive 1, the R1 simulation area on drive 1, the F1 simulation area on drive 3 (3344 only), or the R1 simulation area on drive 3 (3344 only).

IPL from drive 3 provides operation of the system entirely from fixed media. Also, it enables drives 1 and/or 2 to be used as offline devices when there are no simulation areas on drives 1 and 2 currently in use. In this case, it is recommended that all 5444 unit codes (F1, F2, R1, R2) be assigned to simulation areas on drives 3 and 4.

Choosing any of the disk program load positions results in a specific simulation area being assigned the F1 or R1 unit code. This assignment overrides any assignment made for that unit code during system generation. The assignment of this unit code is made for all partitions; this provides a common system pack for all partitions. Following are the simulation areas assigned to the four switch positions:

Switch Position	Area Assigned
Disk 1 F1	D1A
Disk 1 R1	D1B
Disk 3 F1	D3A
Disk 3 R1	D3B

Note: On a 2-, 3-, or 4-drive 3340 system, IPL is performed only from drive 1.

After the IPL device is selected and the Program Load button is pressed, the date prompt appears on the CRT screen. The date is entered in the format chosen during system generation (either mm/dd/yy or dd/mm/yy). If the system date is not given, it must be supplied before any program execution is initiated. The date is then supplied with a DATE OCL statement or a DATE command. The system date is used for time stamping certain messages by the time-of-day macro instructions, the RPG II TIME operation code, and by the COBOL and FORTRAN CFTOD modules. The system date can be changed by the operator only with the DATE command and only when all partitions are at end of job. If the interval timer is specified at system generation, the system date is changed at midnight, when the timer changes from 23.59.59 to 00.00.00.

After the response to the system date has been processed, the CRT display contains EJ messages for all the partitions. At this point, there are three possible actions:

- If spooling was not chosen during system generation, the operator can now initiate program execution by responding to the EJ message for the partition in which the program is to execute.

- When spooling is chosen at system generation, the spooling support is loaded with the supervisor during IPL. If spool is not going to be used, the operator has the option to cancel spooling support at this point. The CANCEL SPOOL command causes the spooling routines to be removed from main storage, making additional main storage available to any partition. (You must perform another IPL when spooling support is desired.) The operator can now initiate program execution by responding to the EJ message for the partition in which the program is to execute.

- When spooling is to be used, the operator can start spooling either by responding to an EJ message for a partition or by entering a START SPOOL command.

  Activating spool automatically by initiating execution in a partition causes space for the spool file to be obtained on a main data area according to the disk unit and size specified during system generation. Any existing spooled data remains in the spool file and is not lost.

  Activating spool with a START command is required only to override the disk unit and/or size of the spool file specifications made during system generation or to remove any existing spooled data. After the START command has been entered and processed, the CRT display contains SPOOL IS ACTIVE messages for each partition being spooled. After jobs have been placed on the spool reader queue, the operator can initiate program execution by responding to the program message for the partition in which the program is to execute. (See *IBM System/3 Model 15 User's Guide to Spooling,* CG21-7632.)

  On 5704-SC2, there are generally more messages on the CRT than on 5704-SC1. If all active messages do not fit one CRT screen, it may be necessary to scroll through all messages by repeatedly pressing the PF12 key.

At this point, it may be helpful for the operator to display the system status to determine that the system is set up as desired. The DISPLAY STATUS command provides the system status on the CRT screen.

PROGRAM EXECUTION

A minimum of two OCL statements are required to load a program into the system for execution. LOAD and RUN are the two basic OCL statements needed to load and execute programs; for those programs in which no disk files are used, they may be the only OCL statements. The LOAD statement identifies the program to be executed and indicates whether the program is to be loaded from an object library or from the partition's system input device. One LOAD statement is required for each program executed.

The following examples show the OCL statements needed to load two system service programs: disk initialization and file delete.

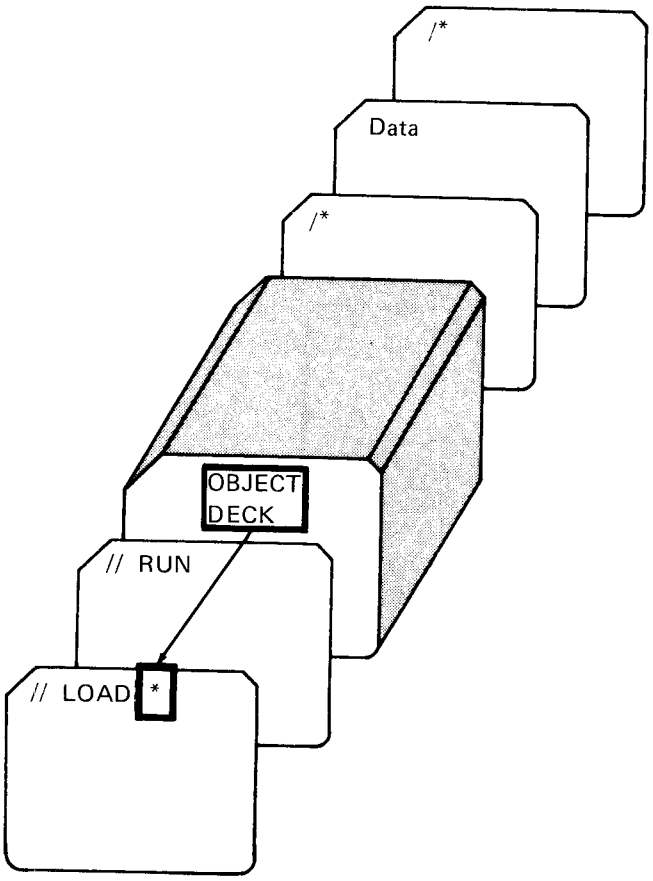
```
// RUN
// LOAD $DELET,F1
```

The file delete program is loaded.

```
// RUN
// LOAD $INIT,F1
```

The disk initialization program is loaded.

To load an object program from the system input device, an * must follow the word LOAD. (The * tells the system that an object deck follows the RUN statement.) A /* statement must follow the object deck; any data will follow the /*. The following example illustrates the statements required to load a program from the system input device:



The object program is temporarily copied into the object library on the system pack and then loaded into main storage for execution. LOAD * programs will be accepted in any partition; 5704-SC2 permits several partitions to execute non-overlay LOAD * program simultaneously.

Job and Step Processing

The input stream (programs to be processed) is made up of two different units of work: jobs and job steps. To support these two units of work, the system provides two processing modes: job mode and step mode. Job mode is indicated to the system by the presence of JOB statements in the input stream. Step mode is indicated by the absence of JOB statements in the input stream (see Figure 2-3). Both modes can be used in the multiprogramming environment (two or more partitions running concurrently) or when only one partition is in use. One partition can be running in job mode while another is in step mode. Only job mode can be used in partitions with spooling active.

Job Step Mode

A *job step* is a request for execution of a program and includes a description of the system resources required by the program. A job step may or may not be part of a job. When a job step is part of a job, the system is in job mode (indicated by the presence of a JOB OCL statement). When no JOB OCL statements have been encountered in the input stream in which job steps are running, the system is in step mode. In step mode, each set of LOAD/RUN statements is processed independently. The system input device is released after the RUN statement has been processed and the program is executing. Step mode is not supported in partitions that have spooling support active.

Note: Some programs, such as the RPG II compiler and disk sort, require system input device dedication while executing, in which case, the system input device remains assigned to that partition until those programs release it. An attribute bit in the object library directory entry indicates whether or not a program requires system input dedication.

When a 2 or 3 option is selected for a system message and the system is in step mode, only the current job step is canceled. The system flushes the input stream until a JOB, LOAD, CALL, /., or /& statement is encountered. If a JOB, LOAD, or CALL statement terminates the flush, the system input device stays assigned to the partition.

The LOAD statement is usually the beginning of a job step's OCL; however, the beginning may be indicated by one of the following:

- The first record of a job's OCL.
- If the previous job step was canceled and the flush ended on a /. or /&, the record after the /. or /&.
- If the previous job step was canceled and the flush did not end on the /. or /&, the statement that ended the flush.
- If the previous job step was not canceled and the partition is not in a nested procedure, the record after the last record read for the previous job step.
- If in a nested procedure, the statement after the CALL to a procedure with a LOAD statement (end of procedure goes to the statement after the CALL in a nested procedure or, if at the end of a procedure nest, the next record in the input stream).

The statements following this record (the beginning of a job step's OCL), through the next RUN statement and any records read from the input stream by this step, are associated with this job step. In job mode, any immediately following /. or /& statement is also included. If the step is canceled in job mode, the end of the step's OCL is the end of the OCL for the job in which it is contained. If the step is canceled in step mode, the end of the step's OCL is the statement before the next /., /&, CALL, LOAD, or JOB statement (the /. or /& statement is included in this step).

Job Mode

A *job* is a group of related job steps that must be executed in sequence. Each job step must be successfully completed before the next job step can be executed. A job includes all the programs, files, and OCL statements necessary to complete the task. The beginning of a job is usually indicated by the JOB OCL statement in the input stream for a partition. However, the beginning of a job's OCL may be indicated by one of the following:

- The first record in an input stream after IPL.
- With input spooling, the first record of the spooled records for a job.
- The first record from the input stream after the last record of the previous job (if the previous processing mode was job mode) or job step (if the previous processing mode was step mode).

All OCL statements that follow this record or the JOB statement but precede the next JOB statement are associated with the job. Any number of LOAD/RUN combinations and/or CALL/RUN combinations can be included in a job. The job name from the JOB OCL statement is used to identify, monitor, and control jobs when spooling.

When job mode is used, JOB OCL statements must be included in the input stream to define the jobs. They indicate to the system that job mode is active and that the system input device assigned to the partition will not be released until job mode is terminated. Processing in job mode ensures that a group of programs are executed sequentially, in the correct partition, and in a partition that has sufficient main storage to accommodate all steps of the job. When a job is initiated, the system checks the PARTITION and CORE parameters, if specified, in the JOB OCL statement to ensure that the job is initiated in the correct partition and that enough main storage is allocated to the partition to satisfy the CORE parameter. If either of these conditions is not met, the system issues a message and the job must be canceled. The system input device, unless it is the CRT/keyboard, must be dedicated to the partition being run in job mode on nonspooling systems, or on systems that do not have input spooling.

To end job mode and return to step mode, a /. OCL statement is placed in the input stream. This statement indicates to the system that job mode is to be terminated and the system input device is no longer dedicated to that job. Job mode will resume when another JOB statement is processed.

Job mode must be used when spooling is active. The spooling routines handle job scheduling and job initiation. Jobs are scheduled for execution based upon the priority specified in the JOB statement. Spooling ensures that jobs are executed in the requested partition, and that all job steps are executed sequentially in the same partition. Jobs that cannot be scheduled for initiation because the required main storage is not available are bypassed until the required main storage is available.

When a 2 or 3 option is selected to a system message and the system is in job mode, all remaining job steps in the job are canceled. On systems that have input spooling, job processing in the partition resumes with the next job on the reader queue. On systems that do not support input spooling, the job stream is flushed until a JOB or /. statement is encountered. If output spooling is supported on the system, spooled output that has been created by the job are saved. If a JOB statement terminates the flush, the system input device stays assigned to the partition.

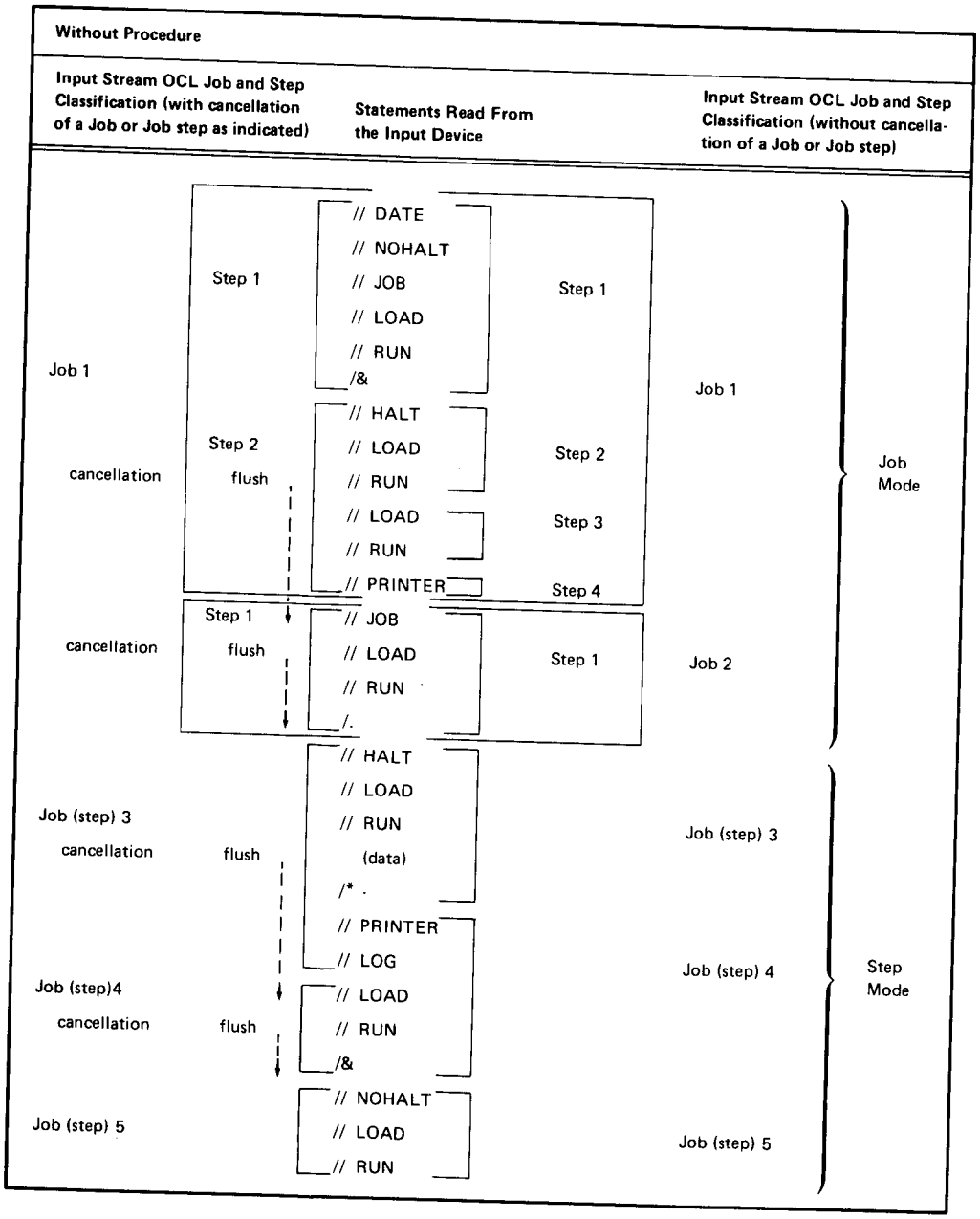


Figure 2-3. (Part 1 of 2). Job and Job Step Classification

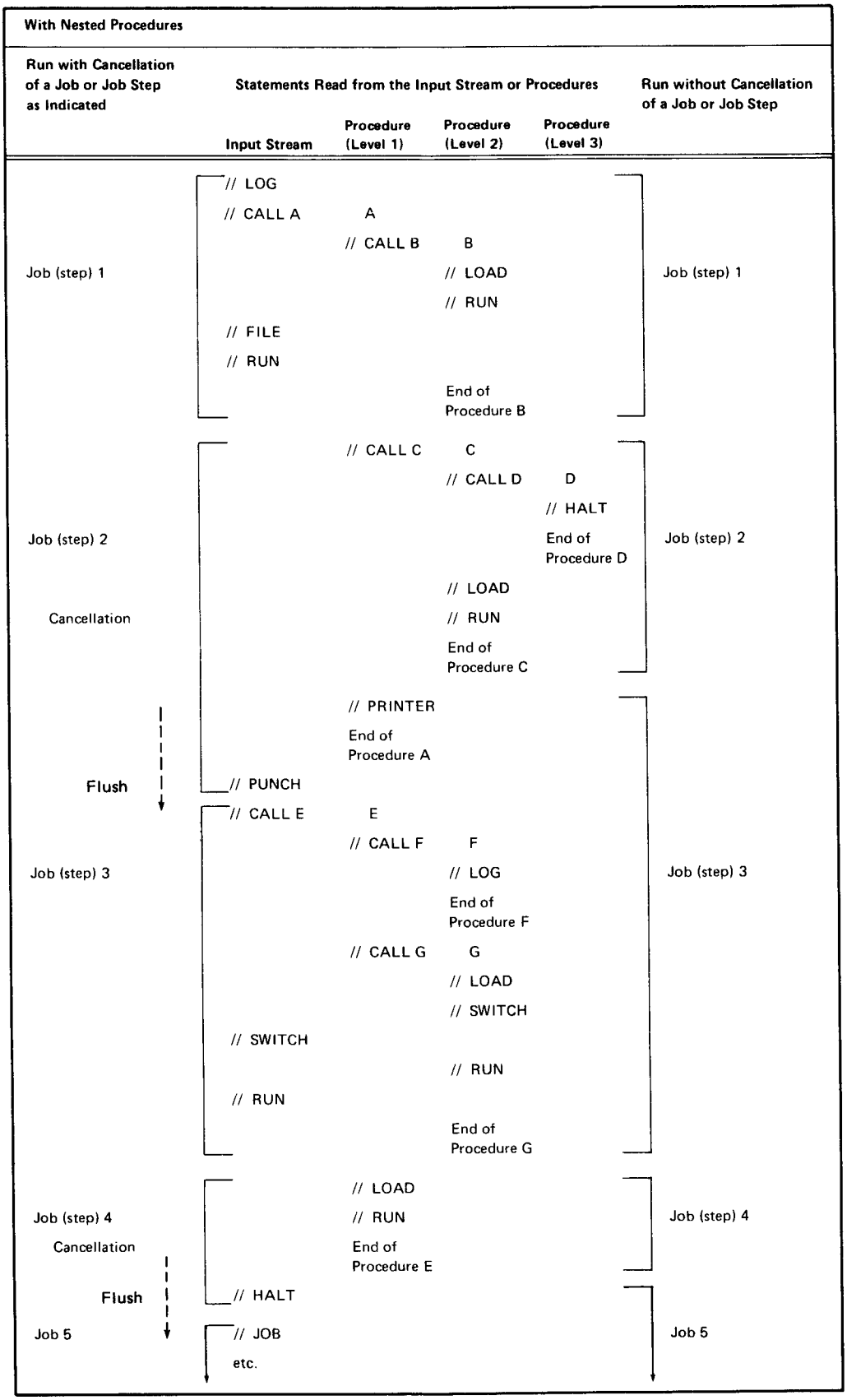


Figure 2-3 (Part 2 of 2). Job and Job Step Classification

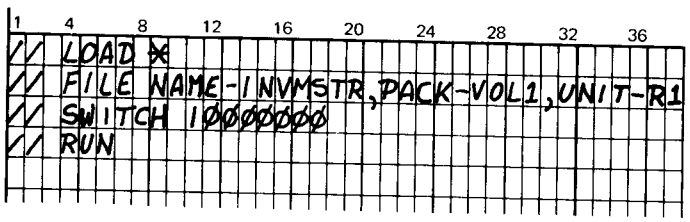
EXTERNAL INDICATORS

External indicators enable a user to influence the execution of a program from a source outside the program itself. Eight external indicators are assigned to each partition; all external indicators are set off during IPL. The SWITCH OCL statement is provided to modify the external indicators for the partition in which the statement is received. When a SWITCH statement sets an indicator, that indicator remains on until one of the following occurs:

- Another SWITCH statement sets it off.
- A JOB statement is received in the partition.
- A /. statement is received in the partition, in job mode.
- Another IPL is performed.

Eight characters in the SWITCH statement correspond to the eight external indicators.

In the following example, a program using one disk file (INVMSTR, an inventory master file) uses the SWITCH statement to set on one external indicator and set off all others.



SYSTEM SEVERITY CODES

A severity code is an indicator to the system specifying when the default option can be used for a system message. When a severity code is not specified on the NOHALT OCL statement, the operator must respond to all system messages except informational messages; this includes informational end-of-step and end-of-job messages.

The severity code can be 1, 2, 4, or 8. The system selects the default option for system messages of a severity less than or equal to the code indicated. Severity codes and defaults are assigned to most system messages and cannot be altered. If the severity assigned to a system message is greater than the severity indicated in the NOHALT statement, the system stops, waiting for the operator's response. If the severity assigned to the message is equal to or less than the severity indicated in the NOHALT statement, the system selects the default option for the system message and processing continues. The severity code does not affect system messages having no default options or requiring operator intervention. Severity code 1 is the least severe; severity code 8 is the most severe. Severity codes are reset to no severity at end of job. For more information on system messages and severity codes, see *IBM System/3 Model 15 System Messages,* GC21-5076.

Note: Halts displayed in the message display unit are not affected by the SEVERITY parameter.

JOB STREAM EXAMPLE

The example in Figure 2-4 illustrates the processing of a job stream and some uses of OCL statements (part 1). The example consists of two jobs with each job containing related job steps. The jobs involve three files: customer, inventory, and transaction. The customer file contains such information as customer names and addresses, total amounts of charges over a period, and total amounts of payments over the same period. The inventory file contains such information as item numbers and descriptions, prices of the items, and the numbers of items in stock. The transaction file contains such information as orders for items, refund orders for items returned, and customer payments. The transaction file is used to update the inventory and customer files.

The OCL statements for the jobs are shown in Figure 2-4. Sets of statements in the figure are numbered. The explanations corresponding to those numbers are given in the following section. Separate job streams are indicated by statements 4 through 7 and 9 through 13.

(1) The LOG statement assigns the 1403 printer as the log device for partition 1. The DATE statement supplies the system date, 10/20/76. The operator can enter the system date by responding to the IPL prompt for DATE or by entering a DATE OCC or OCL before the first JOB, LOAD, or CALL statement after initial program load.

(2) The JOB statement places the system in job mode. The job must be submitted in partition 1 (PARTITION parameter).

There must be 32K of main storage available in partition 1 to execute the job (CORE parameter).

Spooling must not be active (SPOOL parameter). The JOB statement assigns the name BUILD to the job.

(3) In the first job, three programs are being compiled and executed. One transfers the customer file from cards to disk, one transfers the inventory file from cards to disk, and one transfers the transaction file from cards to disk. The OCL statements for the RPG II compiler are in a procedure called RPG. A CALL statement, therefore, is used to instruct the system to read the procedure each time the compiler is run. The procedure is located on the F1 simulation area.

(4) In each of these job steps, the RPG II compiler is executed and creates a program with the name RPGOBJ. This program is placed in the object library on F1.

(5) In each of these job steps, a disk file is created. The names that identify the files on disk are:

Customer file	CUST
Inventory file	INV
Transaction file	TRANS

The date for all files is 10/20/76.

The cards containing the records to be transferred to disk are being read from the same device as the OCL statements. In each case, the cards follow the OCL statements that load the program.

To help identify each job step, a unique name has been placed in each of the LOAD statements. This name follows the // in the LOAD statement.

(6) /& statements are used as a precautionary measure in case the /* statements had not been placed after the source and data cards.

(7) The /. statement ends job mode and acts as a delimiter between jobs.

(8) The second job uses files created in the first job. The first job step compiles a program. The second job step executes the program compiled in the first job step. The third job step sorts a data file.

(9) The JOB statement places the system back in job mode. The name assigned to this job is INVENTORY. Spooling must not be active. The job must be submitted in partition 1. There must be 32K of main storage available in partition 1 to execute the job.

(10) The RPG II compiler is called to compile a program designed to update the inventory file. The program is placed in the object library on F1.

(11) This job step loads the program compiled in step 10 and updates the inventory file. This program can also print the transaction file records. The printer file is conditioned by external indicator 1. The SWITCH statement sets external indicator 1 on, which prevents the printing of the records.

(12) The last job step sorts the newly updated inventory file and writes the sorted file back to the same area on disk.

(13) A /. statement ends job mode.

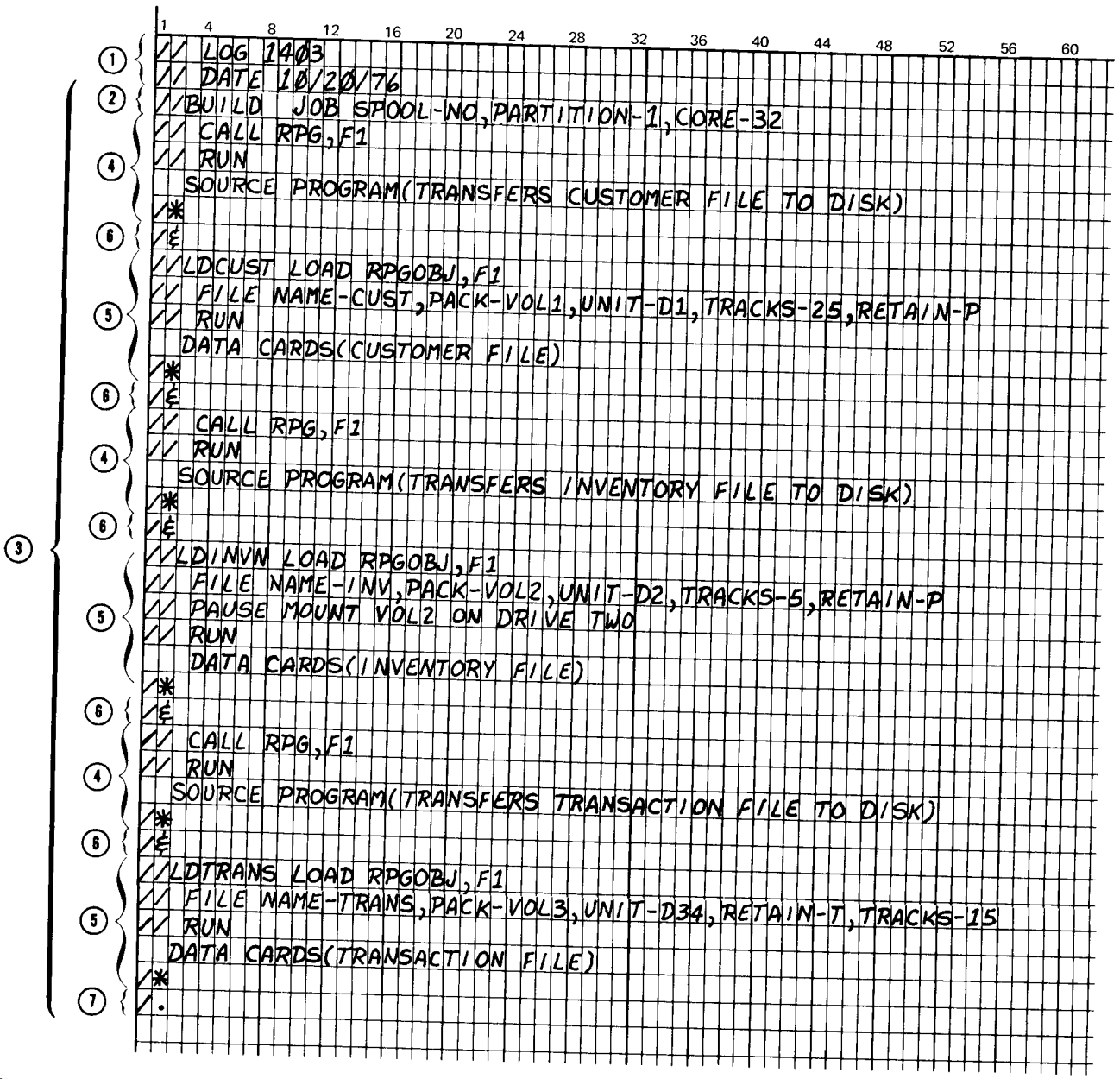


Figure 2-4 (Part 1 of 2). Job Stream Example

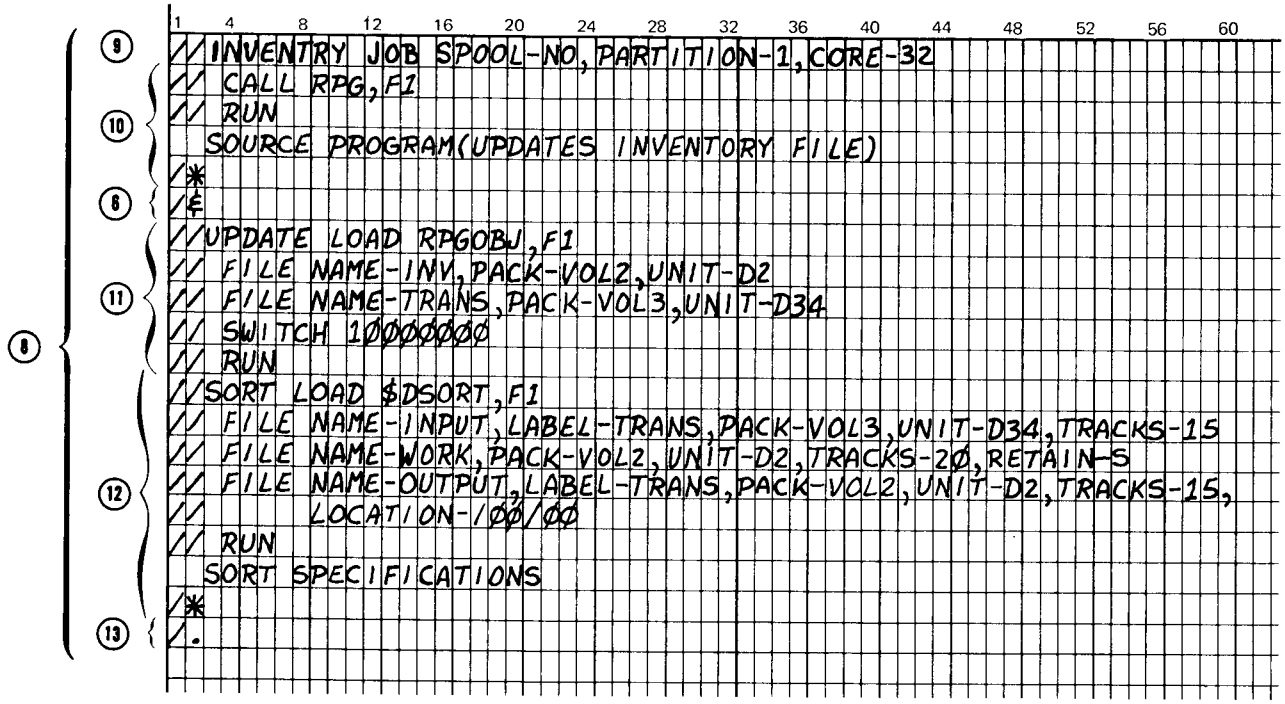


Figure 2-4 (Part 2 of 2). Job Stream Example

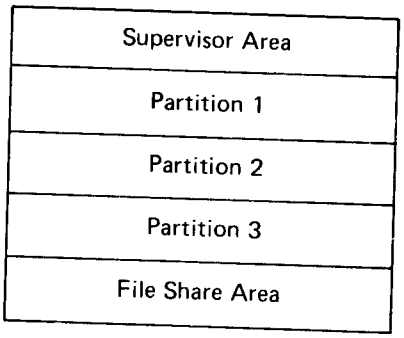
MULTIPROGRAMMING

One of the prime measures of a system's efficiency is the throughput; that is, the amount of work handled in a certain period of time. CPU processing time and its usage are major factors that influence system throughput. *Multiprogramming*, the concurrent execution of more than one program, provides efficient use of the CPU time by allowing programs to share the CPU facilities, thus reducing the time the system is in an unproductive wait state.

With multiprogramming, control is transferred from one task (for example, a user program or a system function such as spool) to another whenever the executing task must await completion of an input or output operation. For example, when a program requests a print operation but the printer is still busy with a previous request, control is transferred to another program. Similarly, when a program requests a record to be read, it must wait until reading has been completed before it can process the data. During the wait, control is transferred to another task. Control is also transferred when a system message occurs. Since most programs wait a significant amount of time for I/O completion, sharing the system facilities reduces unproductive wait time.

With multiprogramming, a disk file may be shared by two or more programs executing concurrently, see Figure 2-1 under *Compatible Access Methods for File Sharing*. This enables programs of various functions to use the same disk file without being concerned about job scheduling. When a particular program requires exclusive use of a disk file, SHARE-NO may be specified on the FILE statement.

Three program partitions are supported by 5704-SC2:



Each partition can contain a separate program, thus allowing concurrent execution of up to three programs. Each program is logically independent, but shares the CPU facilities with the other programs. When CCP is executing in a partition, that partition can have up to 15 tasks executing. For sizes of partitions, see *Program and Partition Sizes* under *Program Facilities.*

The supervisor controls priority for CPU processing. When spool is active, it is assigned the highest priority, with partition 3 second priority, partition 2 third priority, and partition 1 the lowest priority. This priority sequence is set at IPL time but may be changed via a PTY command. The PTY command enables a different priority order to be assigned (such as P2, P1, SP, P3), or one or more of the task can be assigned equal priority (such as SP = P3, P2, P1). A request for equal priorities causes the priority of the specified tasks to be cycled each time the system wait task gets control. For example, when the request is P1 = P2 = P3, the cycle would be *P3, P2, P1, P2, P1, P3, P1, P3, P2,* etc. A request for equal priorities does not guarantee exactly equal priorities since process bound programs will not relinquish control to the system wait task very frequently, but it could have an equalizing effect over a period of time. If one of the partitions contains the communications control program (CCP), it cannot be given equal priority to another task.

When an interrupt occurs, the supervisor gains control, processes the interrupt, and gives control to the highest priority task that is ready to gain control. Control is given up by the high priority task when it encounters a condition that prevents further processing. Control is taken away from a low priority task at the completion of the event for which the high priority task is waiting.

Operating in a Multiprogramming Environment

A job or job step can be executed in any partition. The operator loads a program into a partition by pressing the PF12 key on the keyboard. This moves the cursor on the CRT to the EJ (end of job) message for a partition. When the cursor is positioned for the desired partition, the ENTER key on the keyboard is pressed.

Whenever the EJ message for a partition is responded to, OCL statements are read from the system input device assigned to that partition.

When the system is operating in step mode without spooling support, the system input device assigned to a partition is available for use by any partition after the OCL statements for the current program have been processed. Some programs have input following the OCL and require dedication of that system input device while processing. In this case, the input device is not available until it is released by the program. The following points summarize step mode operation:

- All partitions can share the same system input device; however, only one partition can receive OCL statements and data from the device at any one time.
- Input and output devices required by the programs that are executing must be available. Only the simulation areas, main data areas, and the CRT/keyboard (used as the system input device) can be shared, so careful scheduling of jobs is important.
- The system input device assigned to a partition is available for use by any partition once the current program has released it.

When the system is operating in job mode without spooling support, the following points must be considered:

- OCL statements for all partitions must be entered from separate system input devices (one for partition 1, one for partition 2, and one for partition 3), unless the system input device is the CRT/keyboard.
- Input and output devices required by the programs must be available. Only the simulation areas, main data areas, and the CRT/keyboard (used as the system input device) can be shared by the partitions. Therefore, careful scheduling of jobs is important. The following chart shows what devices can be shared:

Device	Shared by All Partitions
5424 MFCU	No
2560 MFCM	No
2501 Card Reader	No
1442 Card Read Punch	No
1403 Printer	No
Simulation Area	Yes[1]
Main Data Area	Yes[1]
3410/3411 Tape	No
CRT/Keyboard	Yes[2]
BSCA	No
3741 Data Station/ Programmable Work Station	No

[1] The device may be shared; however, there are restrictions on disk files that must be considered.

[2] There are three interfaces to the CRT/Keyboard: system input device, data management, and system log device. The CRT/Keyboard is shared between partitions when the system log device and system input device interfaces are used. The CRT/keyboard is unavailable as the system input device only when it is being used by data management. Otherwise, it is available to any partition requesting input from the system input device.

- The system input device assigned to a partition, except for the CRT/keyboard, remains assigned to that partition for the duration of the job, even if the job step that is executing does not use the system input device. (See *System Input Device* later in this chapter.)

When the system is operating with spool support, job mode is required. Spooling schedules the jobs to be executed in each partition according to the priority assigned to the various jobs and produces the output from the jobs on a priority basis. The priority of a job on the input queue or the priority of the output of a job step can be changed through use of the CHANGE command.

When the system is operating in a spooled environment, the following considerations should be made:

- The operator may schedule jobs for execution in all partitions from the same system input device by supplying the PARTITION parameter on the JOB OCL statement. Once a job has been placed on the spool reader queue, its partition assignment can be changed through use of the CHANGE partition command. (See Part 1 for a description of the JOB statement.)

- The spooled reader, punch, and printer devices may be shared by all partitions when using spooling.

Operator Control Commands for Multiprogramming

The following operator control commands (OCC) provide the user a means to control multiprogramming. (Appendix C contains the format of the commands; for a complete description of the commands, see *IBM System/3 Model 15 Operator's Guide,* GC21-5075.)

- CANCEL — The CANCEL command terminates the job executing in the specified partition. If the canceled program is a job step in a job or a part of a chained procedure, all remaining steps in the job or procedure are also canceled. Processing can continue with the next job.

- CHANGE — One function of the CHANGE command is to alter the partition assignment of the job on the spool reader queue when spool is active. This is done via the PTN operand.

  Another function of CHANGE is to alter the main storage requirements of a job on the spool reader queue (spool must be active). The CORE parameter plus a size specifies the minimum amount of main storage required to execute the largest step in the job. It does not assign main storage to the partition; the SET command is used for this purpose.

  The PTY operand of the CHANGE command allows the priority of a job on a spool queue to be altered. (Spool must be active.) The priority of a job determines when it is executed and when its output is printed or punched.

- DISPLAY — The DISPLAY STATUS command formats the CRT screen with the status of the system. Each page (display) is the status of a partition, the system, or the volume IDs. An ENTER response to each page causes the next partition, system status, or volume IDs to be displayed in sequence: system status, P1, P2, P3, volume IDs, system status, P1, etc. The operator may request a particular page by placing the partition number, an S, or an N under the cursor or by specifying the partition or N on the DISPLAY STATUS command.

  The system display includes the maximum program size (maximum size of a program that can be executed) and the size of the file share area. The partition display includes the location of the simulation areas assigned to that partition.

  The volume ID display includes the names of all the disk areas (main data areas and simulation areas).

  The DISPLAY command also allows the system history area (HISTORY) and the contents of the spooling queues (RDRQ, PRTQ, or PCHQ) to be formatted on the CRT screen.

- DUMP — The DUMP command enables the storage allocated to a partition to be dumped. DUMP can also provide a dump of the total storage area of the system.

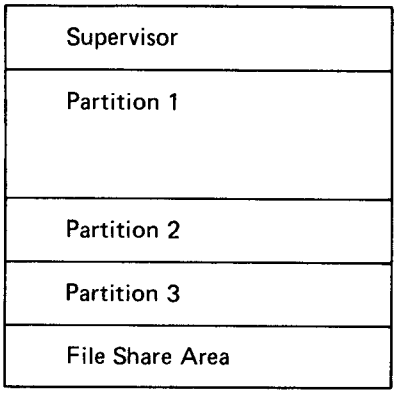
- HALT — The HALT command temporarily suspends processing in a specified partition after the current job step is complete and whenever a system message (except an informational message) is issued in that partition. The HALT command can also cause an informational message when unprinted entries in the system history area are about to be overlaid by additional entries. The HALT command overrides the nohalt mode established at IPL or by a NOHALT OCL statement. Processing resumes when the system message is responded to.

- NOHALT — When a NOHALT command for a partition is specified, the partition does not stop when an end of job or end of step occurs. NOHALT changes the processing mode for the partition from halt to nohalt. It overrides the halt mode established by a HALT command; however, it does not override a HALT OCL statement.

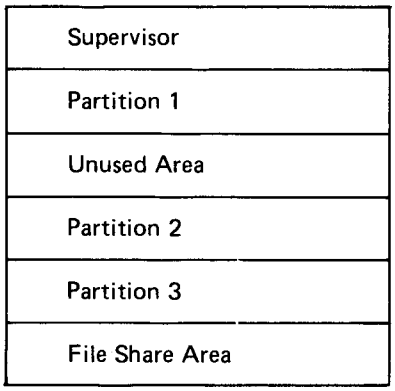
- PTY — The PTY command alters the priority order of spooling and partitions 1, 2, and 3 in the system control program. At IPL, the priority of these tasks is spool, P3, P2, P1, with spool (if active) having the highest priority. The priority of these tasks determines the order in which they receive the services of the system control program. The PTY command also enables one or more of these tasks to have equal priority. Changing the priority may increase the performance of one task; however, it may be at the expense of another task.

- READER — The READER command changes the system input device assigned to a partition. When this command is entered, the specified partition must be at end of job or end-of-job step, and must not have a system input device dedicated to it. The assigned device remains as the system input device until changed by another READER command or READER OCL statement, or until an IPL is performed. The system input device can be changed in a spooled job stream; however, jobs are scheduled for execution from the reader queue only when the system input device is also the spooled reader.

- SET — The SET command overrides the partition and file share area sizes set during system generation or set by a previous SET command.



Before a SET command



After a SET command that changed the size of partition 1. The unused area could be assigned to either partition 2 or 3 with another SET command.

When the SET command is entered to change a partition size, CAN NOT ACCEPT is displayed if the specified partition is not at end of job or end-of-job step. Partitions 2 and 3 both must be at EJ before the size of partition 3 can be changed. If the specified partition is at end-of-job step but not end of job, the partition size may not be decreased to less than:

- The size specified in the CORE parameter of the JOB OCL statement.
- The size of the partition at the start of the job, if the CORE parameter is not specified in the JOB OCL statement.

The minimum partition size for partition 1 is 8K. Partitions 2 and 3 can be set at zero to run the system with only one partition active (dedicated system operation). When partitions 2 and/or 3 are set to zero, the CANCEL and DUMP commands are not accepted for partitions 2 and/or 3. When a partition is set to zero during system generation, the *CT EJ* message will not be displayed for that partition. Both partitions can be reactivated with another SET command specifying 8K or more. The maximum partition size that can be specified is the system storage size minus the size of the supervisor (if partitions 2 and 3 are set to 0) and the size of the file share area (a minimum of 2K). In all other cases, the maximum partition size is the system main storage size minus the supervisor size, the other two partition sizes, and the file share area size. When the size of one partition is decreased to increase the other, the decrease operation must be done first.

When the SET command is entered to change the size of the file share area, all partitions must be at end of job. The minimum file share area size is 2K; larger sizes are specified in increments of 2K.

- START

A START command with a specified partition indicates execution can be started in a partition in which execution was previously stopped by a STOP command. This command can be entered at any time and takes effect immediately.

A START SPOOL command with a specified partition indicates the partition in which spooling is to be initiated. The START SPOOL command specifying a partition may only be entered after the spooling function has been activated at IPL. START is used to reactivate spooling for a partition after the STOP command. If entered at the end of a job when the system input device has been released, spooling for the partition begins immediately. If entered during the execution of a job, it takes effect at the end of the next job that releases the system input device. Once spooling is activated by the START command, it remains active until a STOP command is entered.

- STOP

The STOP command with a partition specified indicates to stop execution in that partition. This command cannot be entered for a partition in which the communications control program (CCP) is executing or when a partition is using certain system resources such as system interlocks. This command takes effect immediately. A START command is required to resume program execution after it has been stopped.

Note: This command can also cause the spool writers to stop execution if they are printing or punching DEFER-NO data from the job step when it is stopped. They will resume printing or punching when execution has been started again in the partition.

The STOP SPOOL command with a partition specified indicates the partition in which spooling is to be terminated. This command can be entered anytime following initiation of spooling at IPL; however, it does not take effect until the end of the next job that releases the system input device. If this command is entered at end of job or when the system input device is released, it takes effect immediately. A START SPOOL command is required to reactivate spool support for that partition.

Multiple Partition Support

The three-partition support provides the capability of operating in the following typical environments:

- CCP–batch–limited application development
- CCP–batch–batch
- CCP–MRJE–batch
- Batch–batch–batch
- Batch–batch–limited application development

Batch processing is the execution of programs such that each program is completed before the next program is initiated. *Limited application development* refers to the compilation and debugging of programs in a batch environment.

Multiprogramming Considerations and Restrictions

Library Maintenance Program

The library maintenance program can execute in one partition while any other nondedicated program, including library maintenance, is executing in another partition. All of the library maintenance functions can be performed in a multiprogramming environment except copying or renaming system entries on the simulation area from which the system is loaded. These functions require the other partitions to be inactive.

Restrictions exist on the functions that can be performed by the library maintenance program if a simulation area is specified that is already in use by another partition. These restrictions are:

- Any delete or rename function (LIBRARY-O, LIBRARY-R, or LIBRARY-ALL) or allocate function that specifies the simulation area from which a program executing in another partition is loaded, will not be performed.

- Based upon the option selected at system generation time or during execution of the configuration record program, the following support applies for the copy function (LIBRARY-O, LIBRARY-R, LIBRARY-ALL, or file-to-library):

  For *catalog to no program packs,* any copy function that specifies a simulation area from which a program executing in another partition is loaded, will not be performed.

  For *catalog to CCP program pack*, any copy function that specifies a simulation area which is currently in use as a CCP program pack, will be performed. If that simulation area is also in use as a program pack for the third partition, the copy function will still be performed. However, if the program executing in the third partition is loaded from a different simulation area, the copy function will not be performed.

  For *catalog to all program packs*, any copy function that specifies a simulation area from which a program executing in another partition is loaded (including the CCP program pack), will be performed.

- Any copy function that specifies a simulation area that is also being used by a system service program in another partition for library purposes will not be performed unless both the copy function and the system service program are performing read only functions.

- If a library function can change the extents of the library (that is, the allocate function, LIBRARY-O, LIBRARY-R, or LIBRARY-ALL copy function, file-to-library function, or LIBRARY-O, NAME-ALL, RETAIN-T delete function), a system service program in another partition cannot use that same area, and files cannot be allocated in that area until the library function is completed.

- Any delete, rename, or modify function that specifies a simulation area that is also being used by a system service program in another partition for library purposes will not be performed.

- Any function that specifies a simulation area that is dedicated to another partition for use by a system service program or for an offline multivolume file or a deferred mount file will not be performed.

Any simulation area that is not in use by a system service program in another partition or has a use not previously described as a restriction can be used by the library maintenance program.

File Share

Program Number 5704-SC2 allows sharing of a disk data file between two or more partitions, tasks, or access of a file in a single program whenever it is logically possible. Disk data management monitors file usage at a block level and allows access for disk *writes* or potential disk *writes* to any portion of the file not in use by another program for the same purpose. *Input only* operations are allowed without regard to block usage by another program. Since potential *write* operations are block protected when file sharing, the greater the block size specified, the greater the portion of the file that cannot be shared at any given period with another potential *write* program.

A potential disk *write* occurs during input/update type operations. When the input block is read, it is protected for a possible update operation. The block is released on an update or the next input request. During sequential and consecutive processing, the block is released when all records in the block are processed. During random processing, the block is released after every record.

Add blocks have considerations similar to input/update type operations. Random adds are protected during the actual add and released so another random add program may access that record. Sequential and consecutive adds are protected during the entire add run and are not available for processing until that program has been completed.

A user may gain exclusive use of a file by specifying SHARE-NO in the FILE statement. SHARE-NO excludes other programs from using a file including input only access methods. When files are not shared between partitions or tasks, SHARE-NO should be used to decrease the disk processing time, especially if large amounts of random adds are to be performed.

When two or more programs are sharing a file, a lockout condition can occur if one of the programs fails to release a block of records and a second program attempts to access a record within the same block. A lockout can also occur when a program is executed in such a manner that another program cannot gain control.

Example A:

1. Program X begins and reads a record to update from file A.
2. Program Y begins and reads a record from file B.
3. Program X attempts to read the same part of file B without updating or releasing file A. It must now wait for program Y to release its block.
4. Program Y attempts to read a portion of file A in the same block as program X without releasing the file B block. Program Y is now waiting on program X and both programs are locked out.

Example B:

1. Program Y performs updates until at some condition it issues a halt after inputing a record.
2. Program X attempts to read and update a record within the same block owned by program Y. Program X will wait for program Y to release the block. Once the halt in program Y is responded to, processing will continue.

Example C:

1. Program Y is reading and updating records randomly from file A. Program Y is coded to loop on a chain operation when the record is not found. It is also a high priority program level.
2. Program X is randomly adding and updating records. It is lower in priority than program Y.
3. Program Y chains for a record higher than any previous records in the file. It loops on the chain operation waiting for program X to add the record. Since data management detects the high key request, no disk operations occur and, therefore, control is not passed to program X, so no further processing is done and the partition must be canceled.

Example D:

1. Filename A and filename B define the same physical file in program Y.
2. Program Y uses filename A to perform updates. The records within the I/O block of filename A are protected until released by program Y.
3. Program Y then uses filename B in an attempt to read and update a record within the same I/O block used by filename A. However, program Y cannot use filename B until the I/O block used by filename A is released.

Because of these considerations, programs that will be sharing files should not be coded to do consecutive reads from different disk files without doing intervening writes. Programs that will potentially hold up usage of a block of records because of length calculations or halts should release the block before the action and reread the block for any update. When this type of coding cannot be avoided, the user must be aware of the possible lockout conditions and schedule jobs accordingly.

During index add processing, if a program executing in one partition is sharing a file with an add program executing in another partition, the key sort is deferred until the last user of the file goes to end of job. If the last user of the file does not have a $INDEX45 or $INDEX40 FILE statement to allocate space for the work file (which is used to sort the keys), the system will attempt to automatically allocate the space. However, the system may not be able to automatically allocate the space. Therefore, if the work file is required, if there is a possibility that the key sort will be deferred, and if sufficient space will not be available for the system to automatically allocate the work file, then the user should include a $INDEX45 or $INDEX40 FILE statement for the sharing program even though it may not be an add program.

During random adds to an indexed file, it is possible for two programs to attempt to add the same record. If both programs do so together, the first program will add the record and the second will get an ATTEMPT TO ADD DUPLICATE RECORD message.

This page intentionally left blank.

Programs coded to perform two random updates to the same disk file without an intervening read will receive disk error messages. Once a record update is issued, an input operation must be performed before another update is attempted.

When adding records to a file, the availability of the added records to another program depends on the access method adding the record and the access method retrieving the record. The most permissive access method is indexed random add. In order to retrieve records added by another partition, random add access must be used by the retrieving program even though no actual adds will be performed by the program. The following chart indicates record availability.

Sharing Access to Added Records

Records Added by Access: \ Record Availability To Access:	CI CU	DI DU	IS ISL ISU ISUL	IR IRU	IRA IRUA
CA	(3)	(3)			
ISA/ISUA	(3)	(3)	(4)	(4)	(3),(5)
IA/IRA/IRUA	(2)	(1)	(4)	(4)	(1)

(1) Added records immediately available

(2) Record added prior to open available

(3) Added records available after close of adding program

(4) Added records available after key sort at end of job or CCP close

(5) Access method not available until add access has completed

CA Consecutive add
ISA Indexed sequential add or input add
ISUA Indexed sequential update add
IRA Indexed random input add
IRUA Indexed random input update add

Multiprogramming Examples

The following sample job streams assume four jobs are to be executed (Figure 2-4). The system configuration consists of a processing unit with 256K, 5424 MFCU, 1403 printer, 3340 disk drives, and a CRT/keyboard. The system input devices are the 5424 MFCU for partition 1 and the CRT/keyboard for partitions 2 and 3. Before executing these jobs, the operator must decide whether to execute them in job mode or step mode. If job mode is chosen, spool or nonspool must be chosen. When the system is operating in step mode or in job mode without spool, all I/O devices used by the job steps must be available. Careful planning and scheduling of job steps is necessary to make maximum use of multiprogramming and to avoid I/O device conflicts between partitions.

Assuming a nonspooled environment, jobs 1, 3, and 4 (which use the MFCU for reading and punching cards) are scheduled in the same partition to avoid conflict with the MFCU. Since job 3 does not require the printer, it is scheduled as the first job in partition 1 to avoid a conflict with job 2 in partition 2. Job 1 is scheduled as the second job in partition 1; however, to avoid a conflict with the printer, step 2 of job 1 cannot be started until job 2 is finished executing. Any subsequent jobs scheduled in partition 2 would have to avoid using the printer or MFCU because job 4 uses both these devices. Partition 3 is not used in this example.

The OCL statements required for partition 1 would be as follows:

```
// DATE 10/20/76
//JOB3 JOB SPOOL-NO
//HISTORY LOAD PHIST,F1
// FILE . . .
// RUN
//PUNCH LOAD PHORD,F1
// FILE . . .
// RUN
/.
//JOB1 JOB SPOOL-NO
//UPDATE LOAD UPDATE,F1
// FILE . . .
// RUN
// PAUSE START NEXT STEP OF JOB1 AFTER JOB2 EJ
//WRITE LOAD ORDRS,F1
// FILE . . .
// RUN
/.
//JOB4 JOB SPOOL-NO
// LOAD $RPG,F1
// FILE . . .
// FILE . . .
// RUN
source statements
/*
// LOAD RPGOBJ, F1
// FILE. . .
// RUN
/.
```

The OCL statements required for partition 2 would be as follows:

```
//JOB2 JOB SPOOL-NO
//CHECKS LOAD PAYROL,F1
// FILE . . .
// RUN
/.
```

If the same jobs were run in job mode with spooling active, all jobs could be read by one input device (see Figure 2-5). Assume that the system date was entered at IPL. After IPL, the following operator control commands would be entered (see Appendix C for a summary of operator control commands):

START RDR (Spooling begins reading the input job stream and placing jobs in an area of the spool file called the reader queue.)

START PRT

START PCH (Can be entered after the spool reader has terminated.)

The following OCL statements would be placed in the spooled reader:

```
//JOB1 JOB PARTITION-1, PRIORITY-3
//UPDATE LOAD UPDATE,F1
// FILE . . .
// RUN
//WRITE LOAD ORDERS,F1
// FILE . . .
// RUN
/.
//JOB2 JOB PARTITION-2,CORE-20,PRIORITY-4
//CHECKS LOAD PAYROL,F1
// FILE . . .
// RUN
/.
//JOB2 JOB PRIORITY-2
//HISTORY LOAD PHIST,F1
// FILE . . .
// RUN
//PUNCH LOAD PHORD,F1
// FILE . . .
// RUN
/.
//JOB4 JOB CORE-16
// LOAD $RPG,F1
// FILE . . .
// FILE . . .
// RUN
source statements
/*
// LOAD RPGOBJ,F1
// FILE. . .
// RUN
/.
/.
```

Jobs would be executed in the partitions as follows:

JOB1 Partition 1, as specified by the PARTITION parameter in the JOB OCL statement

JOB2 Partition 2, as specified by the PARTITION parameter in the JOB OCL statement

JOB3 JOB4 Defaults to partition 1 because no PARTITION parameter is specified in the JOB OCL statement

Since spooling schedules the use of the spooled I/O devices, scheduling of jobs in the partitions is made easier. Also, more efficient use of the multiprogramming capability of the system is realized when operating in a spooled environment.

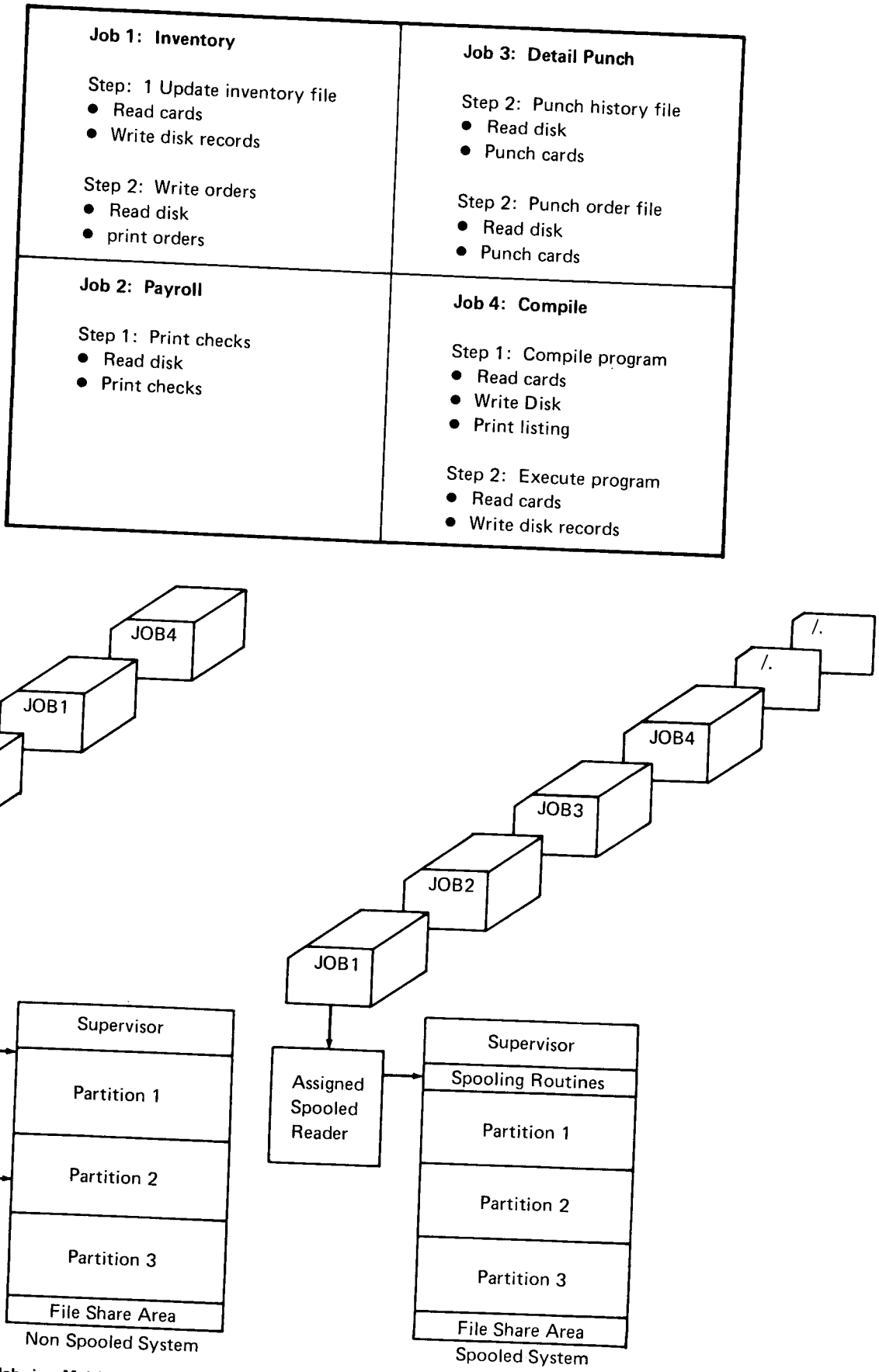


Figure 2-5. Scheduling Jobs in a Multiprogramming Environment

DATE SUPPORT

The System/3 Model 15 uses four date fields: a system date and three partition dates (one for each partition). The dates can be changed by a DATE statement or a DATE command.

System Date

The system date is used for time stamping certain messages by the time-of-day macro instructions, the RPG II TIME operation code, and by the COBOL and FORTRAN CFTOD modules. The system date is set at IPL when the operator responds to the DATE prompt or when the operator enters a DATE command. It is also set when an OCL statement is entered at IPL before any JOB, CALL, or LOAD statement. The system date can be changed by the operator only with the DATE command and only when all partitions are at end of job.

If the interval timer is specified during system generation, the system date is changed at midnight—when the timer changes from 23.59.59 to 00.00.00.

Partition Date

There is a date field for each partition. The partition date is used by programs running in that partition, and it may be different from the system date or from the date in another partition. The partition date is used by the RPG II UDATE field and when creating files.

The partition dates are established initially from the system date; whenever the system date is entered, the three partition dates are also set. The partition date is always restored to the system date at the start of the next job (after EJ).

The partition date can be changed by use of the DATE statement, as follows:

- Job Date: If the DATE statement is entered after the JOB statement and before a LOAD statement, that date will remain in effect for the remainder of that job. It is restored to the current system date for the next job. It is not necessary that the DATE statement precede the first step; it may be placed prior to any step's LOAD statement and will be in effect from that step to the end of that job.
- Step Date: If the DATE statement is entered after a LOAD or CALL statement and before the RUN statement for that step (that is, if the DATE statement is entered within a step), the partition date is changed for the duration of that step. It is restored to the current system date for the next step.

If the interval timer is supported, and if the system date is updated at midnight, then, when the partition date is restored after EJ or ES, it may be different from when the step was started. However, the partition date is changed by the system only after ES or EJ, never during execution of a step.

Use of the DATE statement prior to a LOAD statement ensures that the same date is used for each step in the job. Also, once a DATE statement is entered in this manner, a subsequent DATE statement may be entered for the job, but only if it occurs prior to a LOAD statement.

INTERVAL TIMER

The interval timer provides time-of-day (TOD) support, which maintains the time of day and system date. This information is used to do the time stamping for jobs and job steps and for jobs executed using checkpoint/restart. The time of day is given in hours (00—23), minutes (00—59), and seconds (00—59).

For time stamping jobs and job steps, the system date and time of day (mmddyy or ddmmyy and hh.mm.ss) are used to time stamp the CTEJ and CTES messages. The system date and time of day are saved when the job or job step starts, and the time of day is saved when the job or job step ends. The start time of a job or job step is:

- In halt mode, the response to the EJ or ES message.
- In nohalt mode, after the EJ or ES message is logged in the system history area, but before an attempt is made by the system to read the first record of the next job or job step.

If time of day is not specified during system generation, neither the date nor the time is included in the messages.

Time stamping for checkpoint/restart is as follows:

- When a checkpoint request is received and accepted, the system date and time are saved. This information is put into the CC HY message that is given at a checkpoint.

- When a restart request is received, the system date and time are saved. This information is then put into the CC HW message that is given at a restart.

If time of day support is not specified during system generation, the date and time are not included in the checkpoint/restart messages.

The system date is automatically incremented to the next day at midnight (when the time changes from 23.59.59 to 00.00.00). The partition date, used as the creation date for the disk and tape files, is not updated at midnight until the start of the next job or job step. The partition date is changed by the system only after ES or EJ occurs, never during the execution of a job step. As a result, files created in two adjacent job steps could have different dates. Also, as a result, the CCP date will not change during one execution, even though it may stay up for over a day.

Full interval timer support provides time of day support and also allows the user to set time intervals and to determine elapsed time by using system control programming macro instructions. The five macros—$DATE, $TIOB, $SIT, $TOD, and $RIT—are available and provide the following functions:

- Retrieve the system date and time of day.

- Set the interval timer to cause an interrupt after a specified amount of elapsed time.

- Choose a format for specifying the time intervals.

- Specify the type of intervals to be timed.

- Return the remaining amount of time in an interval.

- Cancel an unexpired time interval.

For information on these macro instructions, see *IBM System/3 Model 15 System Control Programming Macros Reference Manual*, GC21-7608.

SYSTEM INPUT DEVICE

During system generation, a system input device is assigned to each partition. When program loading is initiated in a partition, OCL statements are read from the system input device assigned to that partition. (The system input device is often referred to as sysin.)

The following devices can be assigned as the system input device: 5424 MFCU, 2560 MFCM, 1442 Card Read Punch, 2501 Card Reader, CRT/keyboard, or 3741 Data Station/Programmable Work Station.

When the system is running in step mode, the system input device is assigned to a partition while the OCL statements are being processed, unless the system input device is the CRT/keyboard. After the OCL statements are processed, the system input device may be allocated to another partition. When an error condition occurs during a program and a 2 or 3 option is taken, the job stream in the system input device is read but not processed until a LOAD, CALL, JOB, /&, or /. OCL statement is found. Unless a /. or a /& OCL statement is found, the system input device is not released.

Note: Some programs, such as the RPG II compiler and disk sort, require system input device dedication while executing, in which case the system input device remains assigned to the partition until those programs release it.

When the system is running in job mode and spooling is inactive, the system input device is assigned to a partition for the duration of a job, unless the system input device is the CRT/keyboard. Once a JOB OCL statement is processed in a partition, that partition's assigned input device is assigned to the partition until the job is terminated by a /. statement or a 3 option is taken to a reader hardware error message.

When the system is running in job mode with spooling active, one input device is assigned as the spooled reader from which all input can be read for all spooled partitions. The spooled reader should be the same device as that assigned as a partition's system input device.

Job mode does not cause the system input device to be dedicated when it is the CRT/keyboard.

When nohalt mode is used with job mode, the system input device is not released until halt mode is established and end of job occurs.

The assigned device remains the system input device until changed by a READER OCL statement, a READER command, or until an IPL is performed. The READER OCL statement is read by the current system input device and specifies the new system input device. The READER command specifies not only the new system input device, but also the partition it is assigned to. When the READER command is entered, the specified partition must be at end of job or end of job step, and, if executing in job mode, the partition must have released its current system input device.

The system input device can be changed in a spooled job stream; however, jobs are scheduled for execution from the reader queue only when the system input device is the same as the spooled reader device.

The system control program support for sysin is used by most of the system functions that read input. This includes the system service programs and the language translators. The user may use sysin via the ACCEPT verb in COBOL or a macro statement with the basic assembler. Spooling support does not use sysin.

SYSTEM LOG DEVICE

The system log device, specified during system generation, is used by the system to communicate with the operator and to record messages, OCL statements, and OCC commands.

The device specified as the system log device remains in effect until a LOG OCL statement alters it; the new system log device remains in effect until another LOG statement is processed or another IPL is performed. A system log device is assigned for each partition; for example, the 1403 printer may be the log device for partitions 1 and 2, and the CRT/keyboard may be the log device for partition 3.

In selecting the system log device, the user should consider the following points:

- System messages (informational and decision type) are logged to the CRT independent of the assignment of the log device. The only effect of a LOG CONSOLE statement is to stop logging to a printer.
- System messages, OCL statements, and control statements are logged directly to the 1403 when the 1403 is assigned as the system log device, and print spooling is neither active (intercepting print requests) nor using the 1403 (printing spooled output) and the 1403 is not being used by a program in another partition.
- System messages and control statements are logged to the 3284 when the 3284 is assigned as the system log device and is not being used by a program in a partition.
- When 1403 printed output is being spooled, spooling is active (intercepting print requests for this partition), and the system log device has been assigned to the 1403 printer, all system messages that would normally be logged on the 1403 are placed on the spool print queue. When the spooled output is printed, the system messages are printed along with the job step's output.
- All system messages and OCL statements are logged to the system history area (SHA) on the system pack.

  Note: The SHA may be printed or copied to a device independent file by the system history area display program ($HIST), or displayed on the CRT by use of the DISPLAY command. See Part 4 for a description of $HIST or Appendix C for a summary of the DISPLAY command.

The following information is not logged on the printer but is logged in the system history area (SHA):

- Responses to ERP (error recovery procedures) messages
- Responses to EJ/ES messages
- Volume recognition facility messages and responses
- Spooling messages and responses
- OCC
- OCC diagnostics
- Any ERP message received while a printer ERP message is active.
- Partition identification on OCL statements and control statements for system service programs.

If logging of the above information is required, the log device is assigned only to the CRT, and the contents of the system history area are periodically printed by $HIST.

SYSTEM PRINT DEVICE

The system print device, specified at system generation, is used by various system functions, such as the library maintenance program, to print their output.

The PRINTER statement is used to change the system print device or provide information about printer output that has been spooled.

The PRINTER statement defines the system print device for the partition in which the statement is processed. Either the 1403 or 3284 printer can be defined as the system print device; however, spool supports only the 1403. The system print device may be changed in a spooled job stream; however, if the new device is the 3284, printed output for the system print device is not placed on the print queue. Spooling of system printed output resumes when the system print device is reassigned to the 1403. For more information on the PRINTER statement, see Part 1.

SYSTEM PUNCH DEVICE

The system punch device is used to provide optional output from system functions such as the overlay linkage editor and output from the library-to-punch function of the library maintenance program. A default system punch device may be established for each partition during system generation.

The following devices can be assigned as the system punch device: 5424 MFCU, 2560 MFCM, 1442 Card Read Punch, or 3741 Data Station/Programmable Work Station. The PUNCH statement is used to change the system punch device or provide information about punch output that has been spooled.

The system punch device may be changed in a spooled job stream; however, if the new system punch device is not the spooled punch device, punched output for the system punch device is not placed on the punch queue. Spooling of this punched output resumes when the system punch device is reassigned to the device designated as the spooled punch device.

SYSTEM HISTORY AREA

The system history area (SHA) is a space on the system pack that contains a log or audit trail of the following information:

- OCL statements logged by the system
- OCL diagnostics issued
- Control statements logged by the system for system service programs
- Operator control commands entered
- Diagnostics issued for operator control commands
- System messages issued
- Volume recognition facility (VRF) messages
- Operator responses to system messages
- Unit record restart and 3340 restart messages
- Program product messages
- Display screen images (optional)

A system history area is created on the system pack during system generation; a size option with a minimum of two tracks (48 sectors) is provided. The allocate function of the library maintenance program reserves space for the system history area when allocating libraries on a pack that will be used as a system pack—there is only one system history area per simulation area.

The library maintenance program is the only means to change the size of the system history area (HISTORY parameter of the allocate function) after system generation. The minimum size is 2 tracks; the maximum is the number of contiguous tracks available on the pack following the object library. The following are considerations for determining the size of the system history area:

- Each record in the area is 40 bytes; this is the length of one line on the CRT.
- Each operator control command and each OCC diagnostic requires one record.

- Each OCL statement and each control statement require no more than three records, but generally require only one record.

- Each message and each message response generally require only one record.

The following can be used to determine the number of tracks required:

Number of Tracks	Number of Records
2 (minimum)	233
3	377
4	521
5	665
10	1385
15	2105
20	2825

The system history area is a reusable area. When the space is filled, the oldest entries are overlaid by the newest entries, a process called *SHA wraparound.* To prevent data from being lost, the HALT command with the SHA parameter is used. HALT SHA causes a decision-type message to be issued when the SHA wraparound warning point is reached. (The SHA wraparound warning point is defined as a specific number of tracks that remain in the SHA before unprinted entries are overlaid.) The message is not issued again until either $HIST or $HACCP has been executed. The HALT SHA, CCP command automatically invokes $HACCP when the SHA wraparound warning point is reached. The HALT SHA command remains in effect until a NOHALT SHA command is entered, an IPL is performed, or the SHA is full message is responded to without subsequently executing $HIST or $HACCP.

Note: The system history area copy program ($HACCP) can be used to copy the current portion of the SHA to a disk file. This program can be automatically invoked if the communications control program (CCP) is executing. For information about the system history area copy program, see the *IBM System/3 Communications Control Program System Reference Manual,* GC21-7620.

The library maintenance program ($MAINT) is used to change the size of the system history area. The SHA wraparound warning point in the configuration record is reduced during IPL if it exceeds one-half the new size of the SHA.

Information in the system history area can also be lost when the library maintenance program is used to reorganize an object library (unless compress in place is used), or when the program allocates or changes the size of a source library. Therefore, it is advisable to print the contents of the system history area on a regular basis—as the last step of a job, for instance.

The system history area can be printed or written to a device independent data management file by the system history area display program ($HIST), written to a disk file by the system history area copy program ($HACCP) under CCP, or displayed on the CRT by use of the DISPLAY command. A copy of the system history area can aid in troubleshooting problems and can provide an audit trail of system activity. See Part 4 for a description of $HIST and Appendix C for a summary of the DISPLAY command.

SPOOLING

Spooling support for Model 15 is an optional facility of the system control program for handling unit-record input and output at disk I/O (input/output) speed, thereby increasing total system throughput. Spooling support must be specified during system generation; various options enable spool to be tailored to the user environment.

To handle a system's unit-record I/O without being dependent on I/O device speeds, spooling support builds a file on one of the main data areas and uses this file for intermediate storage. This area is logically divided into queues for reader input and print and punch output data.

As job streams (including OCL statements, programs, and data) for the individual partitions are read, the spool reader routine stores each job on the reader queue. After a job has been completely read and stored in the reader queue, it is ready to be transferred to the appropriate partition (1, 2, or 3) for execution.

When the operator initiates execution in a partition by responding to the EJ message, spooling support presents all the records for a job to the partition as though they were just read from the physical input device. When all job steps for a job have been executed, the spooling support removes the job from the reader queue.

As a job executes, its printed and/or punched output is stored temporarily in the spool file on print and punch queues. The output may be printed or punched later while other jobs are executing, thus allowing a job to execute without having to wait for the unit-record devices.

Whenever a spooled reader, card punch, or printer becomes inoperative, the system continues processing jobs already on the reader queue. When the I/O unit becomes operative again, reading, punching, or printing continues without a loss of output or processing time.

With spooling, jobs can be executed in all three partitions without the need for separate unit-record devices for each partition. For example, one card reader can be the system input device for all three partitions; three card readers are not necessary. Also, one printer can service all three partitions without device contention.

Execution of the spooling functions is controlled with operator control commands and operation control language. For further information on spool support, see *IBM System/3 Model 15 User's Guide to Spooling,* GC21-7632.

CHECKPOINT/RESTART

Checkpoint is a means of recording the status of a user program at desired intervals. Restart is a means of resuming the execution of the program from the last checkpoint rather than from the beginning, when processing is terminated for any reason (with the exception of a controlled cancel) before the normal end of job, such as when a power failure occurs and causes an interruption.

When checkpoint/restart is used, the following programming considerations should be kept in mind:

- Checkpoint/restart enables the user to restart a checkpointed program from the last checkpoint taken provided no intervening program executions have taken place.
- The checkpoint/restart support was selected during system generation.
- Checkpoint requests are accepted only in partition 1. Checkpointed programs must be restarted in partition 1. If partitions 2 or 3 are used to execute a checkpoint program, the checkpoint requests are ignored.
- Only programs 48K or less may be checkpointed. A diagnostic is issued when attempting to checkpoint a program larger than 48K.
- Programs using file sharing or external buffers cannot be checkpointed.
- A LOG statement may be required to reestablish the partition's logging device.
- If any I/O requests from partition 1 are being handled by spooling, checkpoint requests are ignored; therefore, spooling should be stopped in partition 1 using a STOP command. See Appendix C for a summary of the STOP command.
- Any I/O devices being used by spooling that will be used by the checkpoint program must be released by operator control commands. (See Appendix C for a summary of operator control commands.)
- If CCP is active in the partition being checkpointed, the checkpoint request is ignored.
- If the interval timer is being used to time intervals for the partition being checkpointed, the checkpoint request is ignored.
- A PRINTER statement may be required to establish the 3284 as the printer. Multifile tape volumes that have been prepositioned and left there by tape allocate (SEQNUM-X parameter on FILE statement) will not be checkpointed or repositioned during restart.
- A READER and/or PUNCH statement(s) may be required to reestablish the spooled reader and punch devices.
- Compilers and the overlay linkage editor should not run in partition 1 as intervening programs with a LOAD * or temporary checkpoint program. When a program is cataloged by the overlay linkage editor, if the checkpoint program is terminated (immediate cancel), the checkpoint program may be deleted before the restart.

Preceding the initiation of the checkpoint program when spooling support is active, a STOP SPOOL command must be entered to terminate spooling in partition 1.

Assume the following job stream was submitted:

```
//JOBCHKPT JOB PARTITION-1,SPOOL-NO
//CHKPT1 LOAD CHKPT,unit
// PAUSE STOP THE SPOOL READER AND PRINT
   AND PUNCH WRITERS

// RUN
```

The JOB statement includes PARTITION-1 and SPOOL-NO. These parameters prevent the checkpoint job from being run in partitions 2 or 3 or, if spooling is active, in partition 1. If spooling is active, it is the operator's responsibility to stop spooling in partition 1 and resubmit the checkpoint job.

The PAUSE statement is a technique to inform the operator that the checkpoint program will be initiated. This statement is not required if spooling is not active.

The following operator control commands should be entered at this time:

```
STOP PRT
STOP PCH or STOP RDR
```

This releases the spooled devices for the checkpoint program.

This page intentionally left blank.

Restart also requires spooling support to be inactive in partition 1, and the devices (same devices as used by the checkpoint program) must be released by using operator control commands.

The following OCL is required:

```
//JOBRSTR JOB SPOOL-NO,PARTITION-1
// LOAD $$RSTR,unit
// RUN
```

SPOOL-NO is required to prevent the job from being run if spooling is active in partition 1. PARTITION-1 is required because $$RSTR can only be executed in partition 1.

Programs can be executed prior to the restart of a checkpoint program; however, the following items should be observed since no system protection is provided:

- The checkpoint program has been permanently cataloged. Also, LOAD * must not be used.
- If any tapes are processed by the basic tape access method (BTAM) or by direct call to tape IOS in the checkpoint program, they are dismounted to allow the intervening program to use the tape units. The user's restart routine must reposition the tapes to the status at the active checkpoint.
- $MAINT is not executed during the time between failure and restart of the checkpoint program.
- The RPG II, FORTRAN, and COBOL compilers and the assembler can be executed if the object program is punched or cataloged to a pack other than the program (load) pack of the checkpoint program. Anything with the same name as the checkpoint program cannot be cataloged on the program pack of the checkpoint program.
- Intervening programs must not access disk volumes in use by the active checkpoint program for new or scratch files.
- Disk files being used by the checkpoint program should not be modified by an intervening program if the user's restart routine stores selected records in the files. This destroys the updates of the intervening program.
- The disk packs online at restart are the same packs as those used by the checkpoint program, at the last checkpoint.
- Intervening and active checkpoint programs must not add to the same file.

  Note: The checkpoint must be deactivated before loading a checkpoint program into partition 1.
- The $DELET function (FORMAT statement) is not used to deallocate space which does not contain files, libraries, or system areas on any packs used by the checkpoint program.

Checkpoints can be taken via linkage provided by COBOL and the basic assembler. See the appropriate reference manual for more information.

INQUIRY PROGRAM

An inquiry program is a program that is identified as an I-type program; it can be either an RPG or FORTRAN program. The program will have attributes of X'4000' after it has been compiled and link edited. I-type programs can only be executed by an inquiry request (pressing the PA1 key on the keyboard). An I-type program cannot be interrupted. I-type programs are most efficient in a multiprogramming environment but can also be used on a dedicated system. Spooling cannot be used for an I-type program.

For more information on executing an inquiry program, see *IBM System/3 Model 15 Operator's Guide,* GC21-5075, *IBM System/3 RPG II Reference Manual,* SC21-7504, and *IBM System/3 FORTRAN IV Reference Manual,* SC28-6874.

SYSTEM INTEGRITY

System integrity and security of user data is provided in several ways:

Data File Security: File labels and volume labels protect data files from accidentally being destroyed when a wrong data module is mounted, a wrong program is loaded, or an attempt is made to write over a valid data file.

Every disk data file is protected by a file label containing file characteristics. Some typical fields in the file label are the file name, creation date, retention status of the file, and file type. The user cannot access or change the file without first checking the file label.

The volume label record defines the characteristics of a main data area or a simulation area. A typical field in the volume label is the pack name.

To use a particular disk file required in a program, the user must provide the necessary specifications by means of OCL statements. The system uses the information given in these statements to verify that the correct pack is mounted and that the required disk file or area is available.

Volume Security: There are times when an entire volume (a main data area and its associated simulation areas) is dedicated to one partition's use. During these periods, any request from another partition for any of these areas will be denied (F/message). Conditions which require volume dedication are:

- Offline Multivolume File active
- $INIT (Type-Force)
- Deferred Mounts
- Dismount pending situations (WP message is on the CRT)

5704-SC2 provides a volume recognition facility (VRF) for maintaining disk integrity and providing ease of use for 3340/3344 attention situations. Primary functions of the VRF are:

- Diagnose improper pack changes and request that the original pack be placed back online.
- Diagnose improper state change (write to write inhibited). Changing the state of the drive to write inhibited cannot be allowed (0B message) if any activity exists on the entire drive.
- Automatic restart for 3340/3344 diagnostic messages and pack change messages. Messages which qualify for automatic restart are listed in the *IBM System/3 Model 15 System Messages Manual*, GC21-5076.

The VRF maintains a table of pack names for each disk area (main data areas and simulation areas). This table exists in resident storage, is initialized at IPL and updated any time a valid pack change occurs or a system service program changes the pack name ($SCOPY, $DCOPY, $COPY, and $INIT). Read/Write state of each drive is also maintained in this table.

Note: Changing a pack name by any other means is not allowed and may cause unpredictable results unless immediately followed by an IPL.

AUTOMATIC MESSAGE RESTART (UNIT RECORD RESTART)

Automatic message restart eliminates the need for system operator intervention in responding (with a 1 option) to certain system messages.

This function of the SCP can be included during system generation by the user specifying the unit record restart and/or the extended restart option(s). Regardless of the option selected, certain messages are automatically restarted for the 3340 Direct Access Storage Facility and the 3344 Direct Access Storage. For more information about automatic message restart, refer to the *IBM System/3 Model 15 System Messages,* GC21-5076.

For automatic restart, the message response is placed in the system history area with an R in the response position (indicating automatic restart and implicitly meaning a 1 option response). If the response is logged to a printer, the response position will contain a 1.

The configuration record program ($CNFIG) can disable or enable the extended restart function if the user previously selects the extended restart and/or the unit record restart function(s) during system generation.

All messages that apply to the automatic restart function are included in the *IBM System/3 Model 15 System Messages,* GC21-5076 publication.

Unit Record Restart (System Generation Option)

Unit record device error recovery messages can be automatically restarted for the following devices: 2501 Card Reader, 1442 Card Read Punch, 5424 MFCU, 2560 MFCM, 3741 Data Station, 3741 Programmable Work Station, first 1403 Printer, and second 1403 Printer.

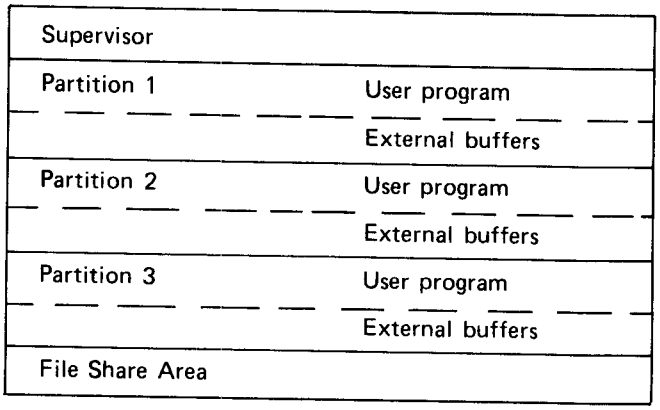
Extended Restart (System Generation Option)

The following messages are automatically restarted when the appropriate resource is available:

Component ID	Halt ID	Subhalt ID	Resource Needed
	F/	Px	Disk
	WA	Px	Disk
SP	UT	NI	Job available on reader queue
SP	UT	RE	Job available on reader queue
	82	02	Tape

MAIN STORAGE USAGE

CPU main storage is logically structured (carved up) during system generation according to specified options describing the user's operating environment. The following illustration shows the logical structure.



The supervisor contains the system control programs that perform the functions necessary for the operation of a system. The size of the supervisor is based on the options chosen at system generation. The approximate main storage requirements for each of the functions is as follows:

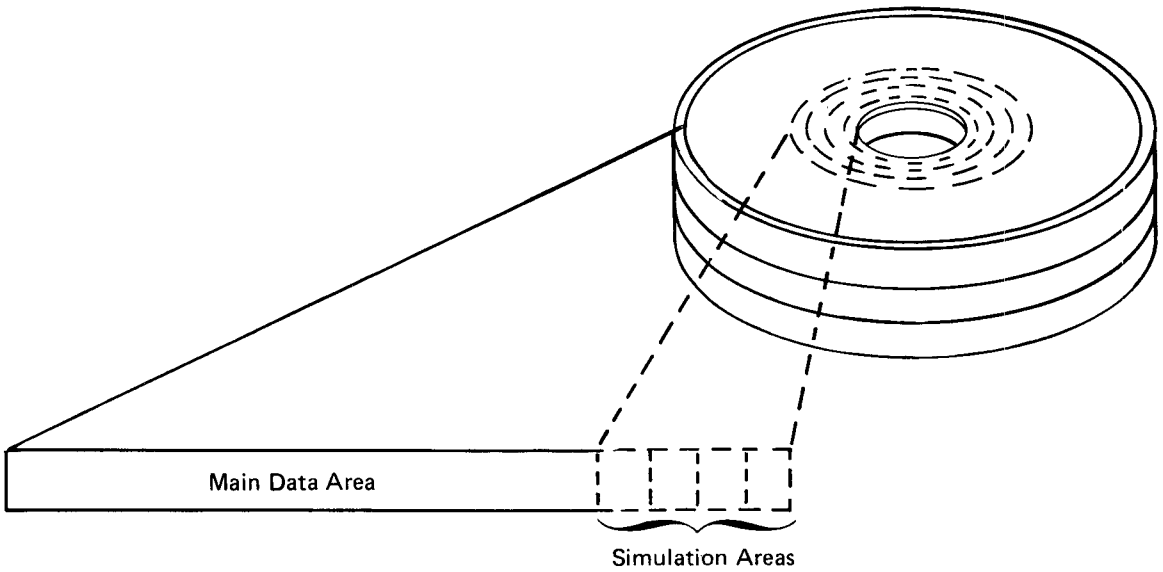
Base supervisor	23.07 (includes 3340/3344 support and and minimum 2K file share common area)
Options:	
File Sharing—additional common area	0-14K
Tape support	1.16
3741 support	0.50
3284 support	0.50
2501 support	0.21
1442 support	0.25
MFCU support	0.65
MFCM support	0.69
I/O storage protection	0.49
Unit record restart	0.34
Support for MLTA/BSCA/LCA/DA	1.41
Support for SIOC	1.81
Memory resident overlays	0.50
Interval Timer:	
Time of day only	0.47
Full timer support	2.00
Spooling:	
Minimum	7.11
Maximum	20.79
CCP:	
Minimum	1.90
Maximum	6.64

Part 3. Disk Storage

Disk storage is provided by one of the following configurations: 3340 Model A2; or 3340 Model A2 and 3340 Model B1; or 3340 Model A2 and 3340 Model B2; or 3340 Model A2 and 3344 Model B2.

3340 DIRECT ACCESS STORAGE FACILITY

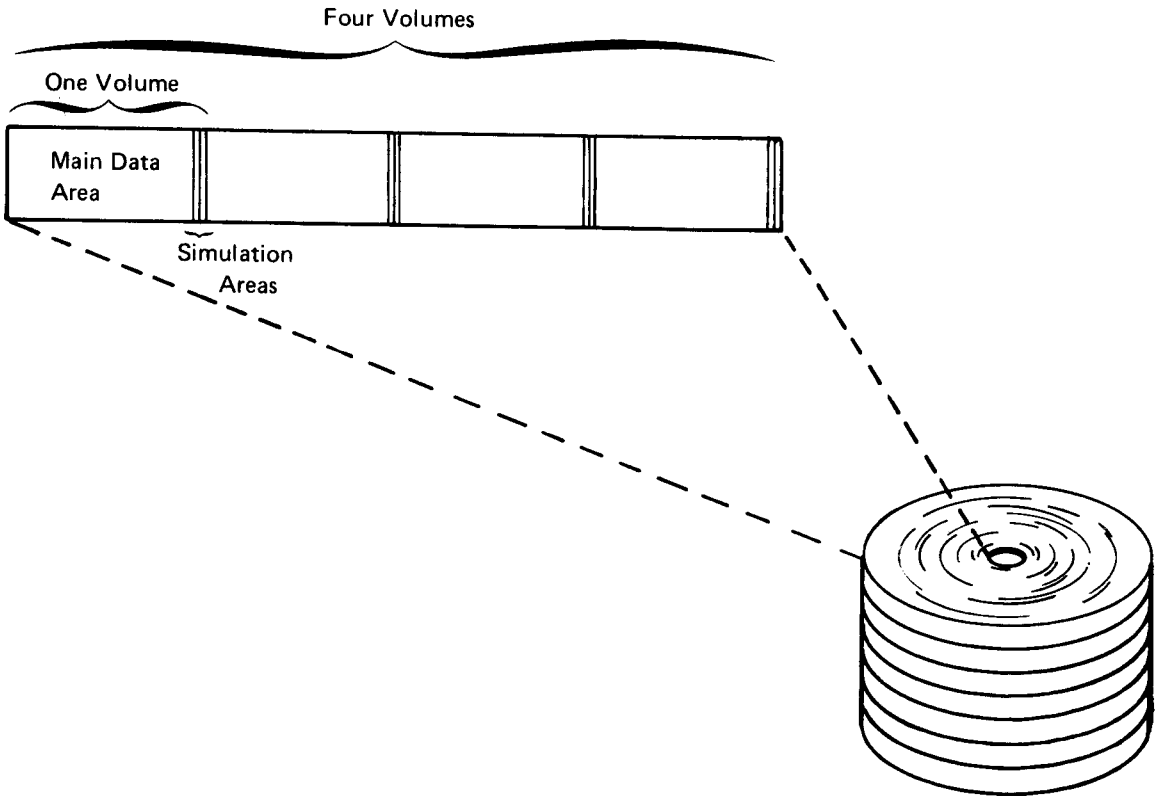
A 3340 Model A2 or 3340 Model B2 consists of two drives. A 3340 Model B1 consists of one drive. Each drive utilizes a removable data module (referred to as a volume) as the storage medium. Each volume consists of a main data area and its associated simulation areas. The following illustration shows the format of a data module (volume) mounted on drive 1 or 2.



3344 DIRECT ACCESS STORAGE

A 3344 Model B2 consists of two drives. The storage medium on each drive is fixed and cannot be removed. The storage medium on each drive is divided into four volumes. The format of each volume is similar to a 3340 volume. One exception is that only two simulation areas are associated with each main data area. Also, the main data area on a 3344 (single volume) has a larger capacity (45.7 million bytes) than the main data area on a 3340 volume (40.8 million bytes).

The following illustration shows the storage medium divided into four volumes. (A volume is one main data area and its associated simulation areas.)



The three parts of Figure 3-1 show the capacity and format for each storage medium used on the disk storage devices. Part 1 shows the byte capacity for each division of the storage medium. Part 2 shows the byte capacity, by model, for the main data areas. Part 3 shows the byte capacity and the cylinder designation for each division of a volume.

DISK CAPACITY IN BYTES	3348 Data Module (3340) Drives 1 and 2	3348 Data Module (3340) Drives 3 and 4	3344 Direct Access Storage[1]	3344 Direct Access Storage[2]
User data area	40,796,160	40,796,160	45,711,360	182,845,440
Simulation area	9,830,400	4,915,200	4,915,200	19,660,800
Subtotal	50,626,560	45,711,360	50,626,560	202,506,240
Cylinder 0 (reserved)	245,760	245,760	245,760	983,040
Subtotal	50,872,320	45,957,120	50,872,320	203,489,280
Alternate tracks (reserved)	491,520	491,520	491,520	1,966,080
Other use (reserved)	98,304	98,304	245,760	983,040
Total physical capacity	51,462,144	46,546,944	51,609,600	206,438,400

[1] One volume.
[2] One drive; four volumes.

Figure 3-1 (Part 1 of 3). Byte Capacity of Each Storage Medium

Configurations	Storage in Millions of Bytes
3340[1]	
One Model A2	81.6
One Model A2 and one Model B1	122.4
One Model A2 and one Model B2	163.2
3340/3344[1]	
One 3340 Model A2 and one 3344 Model B2	447.2

[1] Storage size is for main data areas only and does not include simulation areas or reserved areas.

Figure 3-1 (Part 2 of 3). Byte Capacity (Main Data Area) for Each Model

	System Use		User Data Area		Simulation Areas		Other Use		
	Cylinder	Million Bytes	Cylinder	Million Bytes	Cylinder	Million Bytes	Cylinder	Million Bytes	Total
3348 Data Module	0	0.25	1-166	40.80	169-208[1] (4)	9.83[1]	167-168 209	0.49 0.10	51.47[1]
3344 Direct Access Storage									
Each Volume	0	0.25	1-186	45.71	189-208 (2)	4.91	187-188 209	0.49 0.25	51.61
4 Volumes	—	0.98	—	182.85	—	19.66	—	2.95	206.44

[1] If the data module is mounted on drive 3 or drive 4, only two simulation areas are accessible. The starting cylinder is 189 and the capacity is 4.91 million bytes. Total capacity is 46.56 million bytes.

Figure 3-1 (Part 3 of 3). Byte Capacity and Cylinder Assignment for Each Volume

SIMULATION AREAS

System control program 5704-SC2 requires the simulation of a 5444 disk storage drive. All libraries (for example, user programs, system programs, and procedures) are located in an area on disk called a simulation area. The contents of the simulation area are then used in the same way as a 5444 disk storage drive.

Four simulation areas, each having a unique code, can be accessed on each of the volumes mounted on disk drives 1 and 2. Each simulation area is contained within 10 contiguous cylinders of a volume. (A volume is one main data area and its associated simulation areas.) Therefore, a total of 40 cylinders, 169 through 208, are reserved for the simulation areas on these drives. Disk drives 3 and 4 can be a 3340 or 3344. However, regardless of the type of disk drive, each volume on drive 3 or 4 contains only two simulation areas. These areas, having unique codes, occupy cylinders 189 through 208.

Interchanging Data Modules (3340)

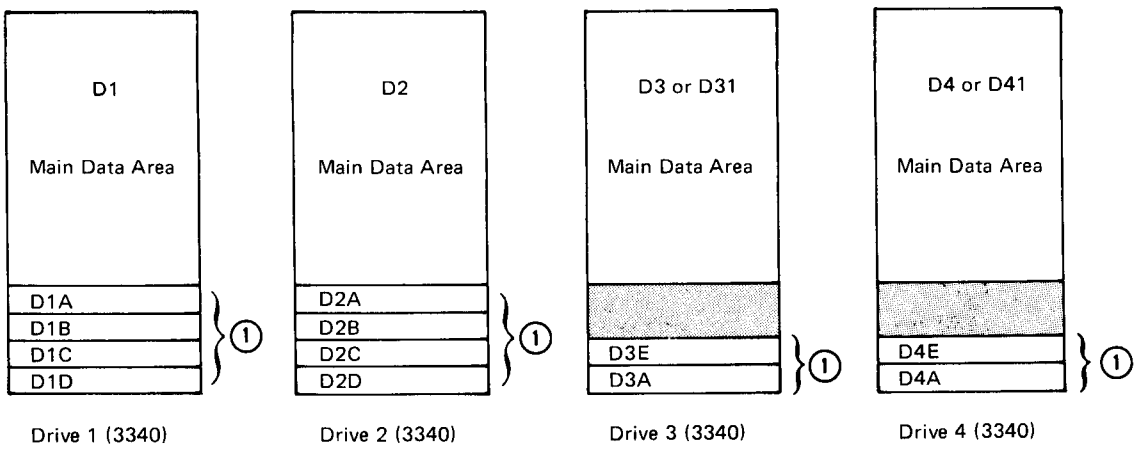
You can interchange[1] any data modules on any drives. However, to access the correct simulation area when a data module is mounted on drive 3 or drive 4 requires special considerations. Refer to Figure 3-2 and note the unique simulation area codes (D3E, D3A, D4E, and D4A) and their location assigned to the simulation areas on drives 3 and 4. Also notice that only two simulation areas can be accessed on each drive.

Simulation areas D3E (drive 3) and D4E (drive 4) occupy the same relative cylinders (189 through 198) as simulation areas coded D1C (drive 1) and D2C (drive 2). Also, simulation areas coded D3A (drive 3) and D4A (drive 4) occupy the same relative cylinders (199 through 208) as simulation areas coded D1D (drive 1) and D2D (drive 2). Therefore, when you move a data module from drive 2 to drive 3 and the system accesses the simulation area coded D3A, the data retrieved is actually from simulation area D2D. This same logic is true for simulation area codes D3E, D4E, and D4A.

The machine will not access cylinders 169 through 188 in the main data area when the data module is mounted on drive 3 or drive 4. Therefore, when you move a data module from drive 1 or 2 to drive 3 or 4, the system cannot retrieve the data located in simulation areas coded D1A, D2A, D1B, D2B.

[1] You must perform an IPL anytime you change the system pack or the pack that contains the active $SPOOL file.

Figure 3-2 shows the format and the unique code for each simulation and main data area when the system has four 3340 disk drives.



(1) Denotes simulation areas associated with each volume.

Figure 3-2. Main Data Area and Associated Simulation Areas

Accessing Simulation Areas

Depending on the disk configuration, the number of available simulation areas can be 8 (two 3340 disk drives), 10 (three 3340 disk drives), 12 (four 3340 disk drives), or 24 (two 3340 disk drives and two 3344 disk drives); they are all directly accessible. However, before a simulation area can be directly accessed, it must first be assigned to a 5444 unit code. The 5444 unit codes are F1, R1, F2, and R2. (The system service program $SCOPY provides access to a simulation area even though it is not assigned to a 5444 unit code.)

Assigning Simulation Areas

Assignment of the simulation areas to the 5444 unit codes must occur during system generation. Assignment occurs when a unique simulation area code, such as D1C, is assigned by partition to a 5444 unit code such as R2. Then, after an IPL is performed, the simulation area D1C is directly accessible when 5444 unit code R2 is specified in the correct partition. For example, assume simulation area D1C is assigned to unit code R2. The following OCL statement causes the system to access simulation area D1C and load the program named $COPY:

```
// LOAD $COPY,R2
```

During system generation, each 5444 unit code (F1, R1, F2, R2) must be assigned by partition to a simulation area code. The simulation area assignment must be unique within a partition. The 5444 unit codes must be assigned in the following sequence during system generation:

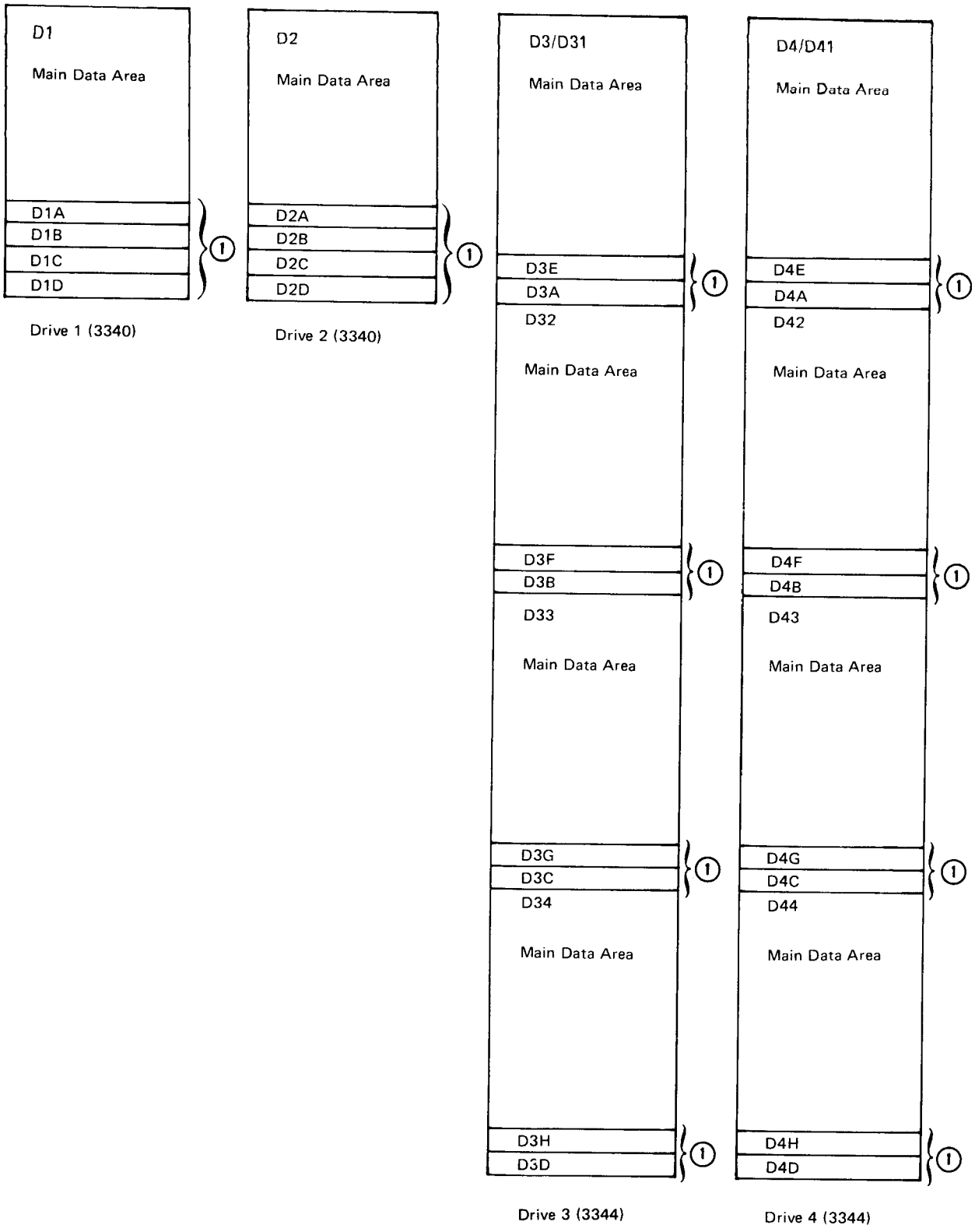
F1, R1, F2, R2

Figure 3-3 shows the format and code for each simulation and main data area when the system has two 3340 disk drives and two 3344 disk drives.

If you do not assign simulation areas, system generation provides the following default assignments:

F1-D1A, R1-D1B, F2-D2A, R2-D2B (all partitions)

For more information about assigning simulation areas, refer to the *IBM System/3 Model 15 System Generation Reference Manual,* GC21-7616.



(1) Denotes simulation areas associated with each volume.

Figure 3-3. Main Data Area and Associated Simulation Areas

The following considerations and restrictions apply when you assign simulation areas during system generation.

- F1 may be assigned to any simulation area except D1B and D3B. However, when the IPL is performed, if the system pack is on drive 1 or drive 3 and the PROGRAM LOAD SELECTOR switch position is either DISK1 F1 or DISK3 F1, the system overrides the previous assignment and assigns F1 to D1A or D3A in all partitions.
- R1 may be assigned to any simulation area except D1A and D3A. However, when the IPL is performed, if the system pack is on drive 1 or drive 3 and the PROGRAM LOAD SELECTOR switch position is either DISK1 R1 or DISK3 R1, the system overrides the previous assignment and assigns R1 to D1B or D3B in all partitions.
- F2 may be assigned to any simulation area except D1A, D1B, D3A, or D3B.
- R2 may be assigned to any simulation area except D1A, D1B, D3A, or D3B.

Simulation Area Reassignment

Simulation areas can be reassigned (by partition) by use of the ASSIGN statement. This statement cannot be used in a procedure; it cannot be placed between a CALL and a RUN statement; and it is effective immediately after being processed.

The format of the ASSIGN statement is as follows:

```
// ASSIGN 5444 unit code-simulation area code
```

One, two, or three simulation areas can be reassigned on each ASSIGN statement unless the PACK and/or AREA parameter is used. In this case, only one simulation area can be reassigned for each ASSIGN statement.

The new assignment remains in effect until another ASSIGN statement is processed or until an IPL is performed, at which time the simulation area assignments revert to those made during system generation, or during the execution of the configuration record program.

Simulation areas can also be reassigned by means of the configuration record program $CNFIG. This program reassigns, by partition, the 5444 unit codes to the simulation areas by changing the configuration record on the system pack from which the IPL was performed. For more information about this program, see *Configuration Record Program–$CNFIG* in Part 4 of this manual.

The following considerations and restrictions apply to reassignments:

- The simulation area from which an IPL is performed cannot be reassigned.
- Duplicate assignments are not allowed within a partition.

Number of Simulation Area Assignments

The maximum number of simulation areas that can be directly accessed is: 10 in a three-partition system; 7 in a two-partition system; 4 in a single-partition system. When an IPL is performed, the system overrides any previous assignments to either F1 or R1 and assigns the simulation area containing the system pack to all partitions. The following example shows how a maximum of 10 simulation areas could be assigned:

Partition 1	F1 assigned to D1A R1 assigned to D4E F2 assigned to D2A R2 assigned to D3A	System Pack
Partition 2	F1 assigned to D1A R1 assigned to D1D F2 assigned to D4A R2 assigned to D1B	System Pack
Partition 3	F1 assigned to D1A R1 assigned to D2D F2 assigned to D3E R2 assigned to D2C	System Pack

The same assignment is allowed in more than one partition. For example, R2 could be assigned to D2B in all partitions. The following example shows the default assignments and also shows how a minimum of four simulation areas can be assigned:

Partition 1	F1 assigned to D1A R1 assigned to D1B F2 assigned to D2A R2 assigned to D2B	System Pack
Partition 2	F1 assigned to D1A R1 assigned to D1B F2 assigned to D2A R2 assigned to D2B	System Pack
Partition 3	F1 assigned to D1A R1 assigned to D1B F2 assigned to D2A R2 assigned to D2B	System Pack

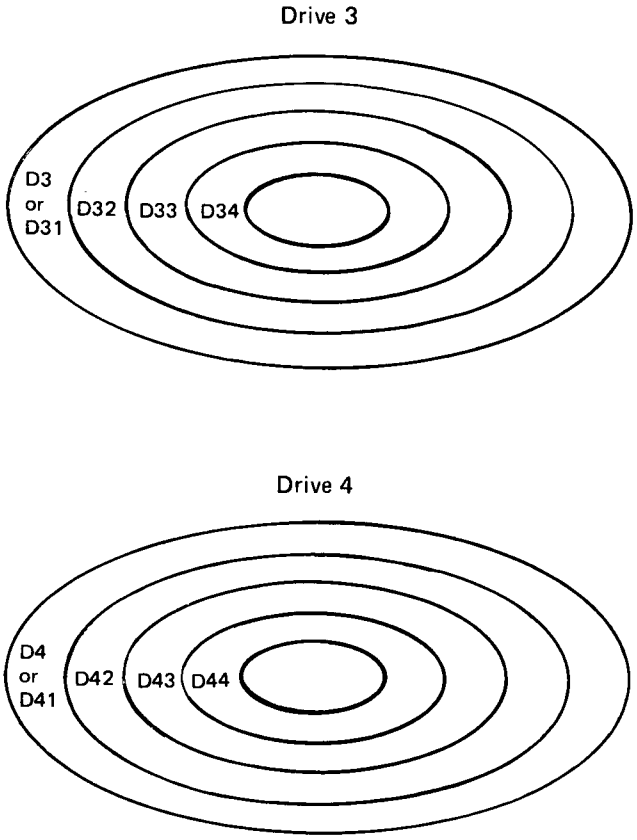
MAIN DATA AREAS

A main data area is a portion of a volume that is used to store data files. (It is not used for library storage.)

Main data areas for each volume start at cylinder 1 and extend through cylinder 166 (3340) or 186 (3344).

Each main data area is accessed by means of a specific code. If the main data area that you want to access is located on a 3340 data module, the disk drive that the data module is mounted on determines the code. For example, if a data module is mounted on drive 1, the code to access the main data area is D1, on drive 2 the code is D2, on drive 3 the code is D3 or D31, on drive 4 the code is D4 or D41.

The storage medium for a 3344 is fixed, and the main data area of each volume is assigned a unique code. The following illustration shows the unique code and the relative location for each 3344 volume:



Disk Space Allocation

For the main data areas on a 3340 or 3344, the allocation of disk space begins at cylinder 1 and extends toward cylinder 166 or 186. If the LOCATION parameter is specified on the FILE statement, file allocation begins at the specified location and extends toward cylinder 166 or 186.

Temporary or Permanent Files

When the LOCATION parameter is not specified, the entire main data area is searched to find the optimum space available for the file. The file is adjusted toward the cylinder 1 end of the space. This procedure is also used to allocate the spool file.

Scratch Files

- Main data area on a 3344:
  - The disk space between cylinders 167 and 186 is searched to find the optimum space available for the file. If sufficient space is found, the file is adjusted toward the cylinder-186 end of the space.
  - If sufficient space cannot be found within cylinders 167 and 186, the entire main data area is searched for the optimum available space. The file is adjusted toward the cylinder-186 end of the space.

- Main data area on a 3340:
  - The entire disk is searched for the optimum space, and the file is adjusted toward the cylinder-166 end of the space.

Considerations and Restrictions

- Use the file compress program ($FCOMP) periodically to remove existing gaps between files.

- To decrease the amount of seek time, group all files associated with a specific job together on the same volume.

- To decrease the amount of seek time, fill all disk space on a main data area of a 3344 to capacity before using a new volume.

ALTERNATE TRACKS

An alternate track is a substitute track selected to accept the contents of a defective track. Forty alternate tracks are available on each volume. For a 3340 volume, they are contained in cylinders 167 and 168. For a 3344 volume, they are contained in cylinders 187 and 188.

An alternate track can be assigned to any track except cylinder 0, head 0 on the following main data areas: D1, D2, D3, D31, D4, and D41. Whenever a program attempts to use a track that has been assigned an alternate, the alternate is used automatically.

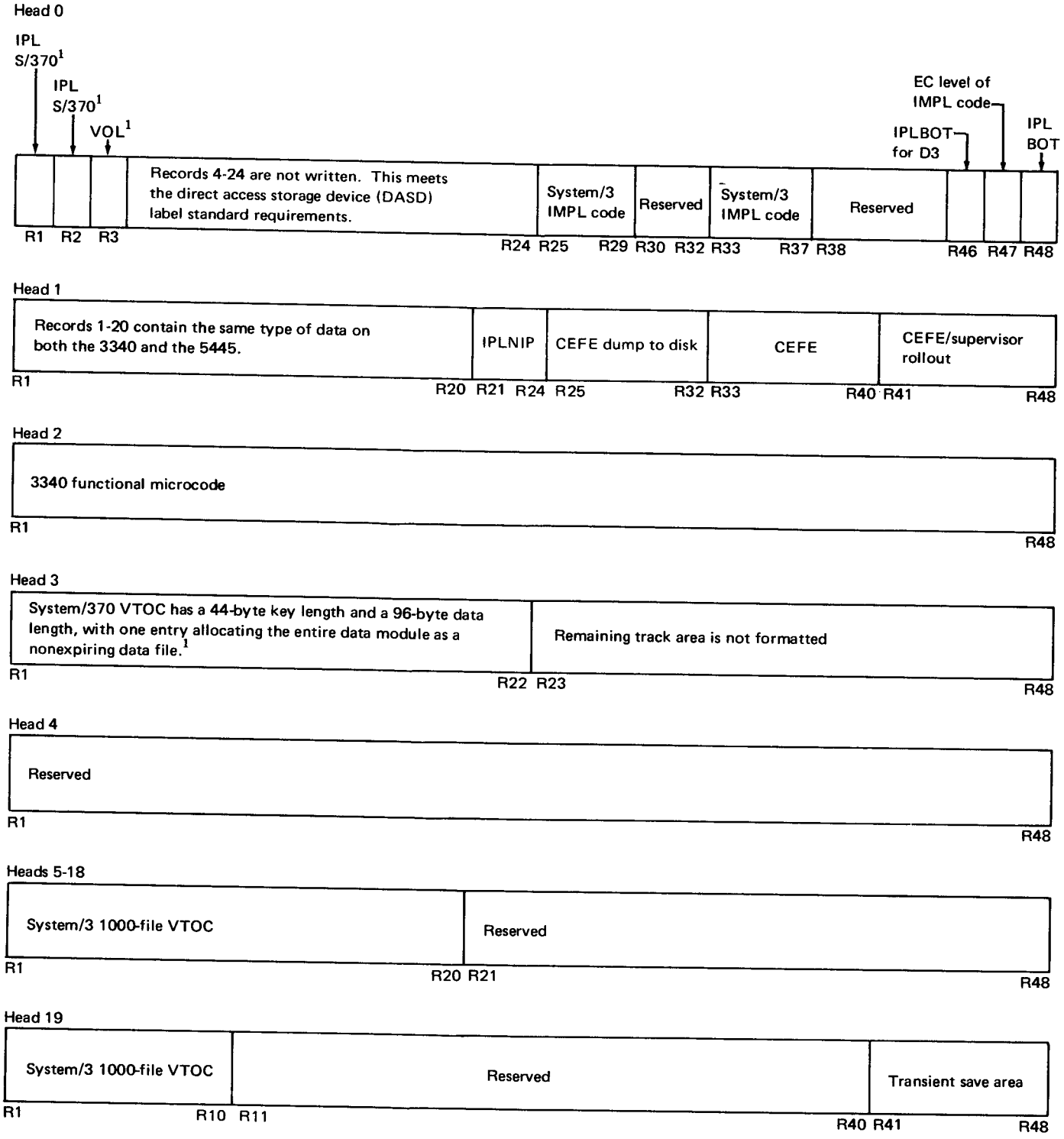
When programs encounter permanent reading or writing errors, the system automatically halts the current operation. The alternate track assignment program can then be executed to test suspected defective tracks; alternates will be assigned as needed. Some of the data might not be recovered when the alternate track assignment program is used. If the data cannot be recovered without an error condition, the record or records are read under reduced hardware checking conditions and are written on the alternate track. The record or records that contain the error are printed in a form that completely identifies the data written on the alternate tracks. This printout should be retained. It is required for running the alternate track rebuild program.

File Processing Considerations

System performance is degraded if an alternate track is assigned for one or more tracks of a file index. It is recommended that the file be moved so that the alternate track contains the data portion of that indexed file.

CYLINDER 0 FORMAT

On each volume, cylinder 0 is reserved for system support of the main data area. It also contains the necesary micro-program and data to perform an IPL for the system. The following illustration shows the format of cylinder 0 and the type of information contained in each record:



[1] These areas are written in count-key-data format (standard data format) readable to System/3 and System/370. Other areas are written in compressed data format.

Considerations and Restrictions

- When using the dump/restore program ($DCOPY) to provide backup for the system pack and cylinder 0, the SYSTEM-YES parameter is specified to dump cylinder 0.
- If you specify SYSTEM-YES when the dump function is performed, you must also specify SYSTEM-YES when the restore function is performed.

INITIAL PROGRAM LOAD (IPL)

The position of the PROGRAM LOAD SELECTOR switch determines which simulation area will be accessed during an IPL. An IPL can be performed from drive 1. An IPL can also be performed from drive 3, provided that drive 3 is a 3344 direct access storage.

The system pack must occupy one of the following simulation areas: D1A (F1) or D1B (R1) on drive 1; or D3A (F1) or D3B (R1) on drive 3.

Figure 3-4 shows the relationship between the position of the PROGRAM LOAD SELECTOR switch and the accessed simulation area.

For more information about IPL, see *Initial Program Load* in Part 2 of this manual.

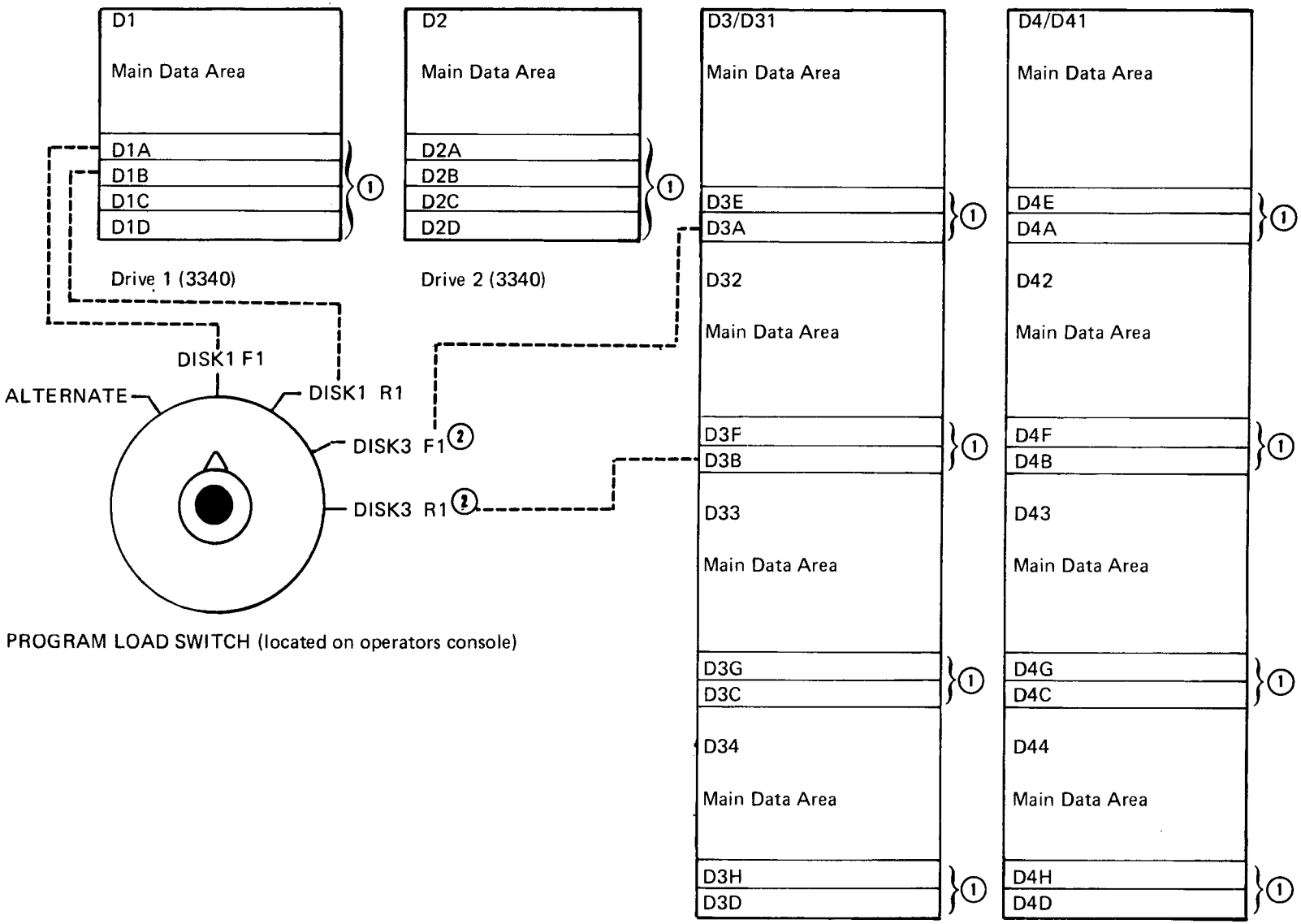


Figure 3-4. Main Data Area and Associated Simulation Areas

(1) Denotes simulation areas associated with each volume.

(2) These positions are not available when the configuration consists only of 3340 disk drives.

Part 4. System Service Programs

The Model 15 SCP (5704-SC2) includes a group of disk-resident system service programs. These programs do a variety of jobs, from preparing disks and tapes to maintaining the system libraries. The following is a list of the system service programs discussed in this section and their function(s):

Program	Name	Program Function
Alternate Track Assignment	$ALT	Assigns an alternate track in place of a defective track and prints the data content of the area in error.
Alternate Track Rebuild	$BUILD	Corrects data on the assigned alternate track.
Configuration Record	$CNFIG	Changes certain information in the configuration record.
Copy/ Dump	$COPY	Copies a file, an entire main data area, or an entire simulation area.
Dump/ Restore	$DCOPY	Provides backup for a main data area or simulation area on magnetic tape, or a simulation area on diskette.
File Delete	$DELET	Deletes temporary or permanent data files.
File Compress	$FCOMP	Moves files for the purpose of removing gaps between files. This program also copies all files from a main data area.
System History Area Display	$HIST	Displays the contents of the system history area.
Disk Initialization	$INIT	Performs surface analysis on disks and formats disks according to disk system management requirements.
Chain Cleaning	$KLEAN	Exercises the IBM 1403 Printer for the purpose of cleaning the print chain (train).
File and Volume Label Display	$LABEL	Prints the information from VTOC pertaining to a single file or to all files.
Library Maintenance	$MAINT	Allows you to produce, maintain, and service libraries.
Spool File Copy	$QCOPY	Copies an entire spool file; copies all or part of a spool print, spool punch, or spool reader queue; copies to the spool reader, spool print, or spool punch queue; copies the status of the spool file queues.
Recover Index	$RINDX	Sorts the file index and updates the VTOC entry of an indexed file to reflect added records in case of abnormal termination.
Reassign Alternate Track	$RSALT	Relocates the alternate track area on a 3340 data module so that the data module can be used on a System/360 or System/370.
Simulation Area	$SCOPY	Supports the simulation areas of a 3340 or 3344 volume.
Tape Initialization	$TINIT	Creates or deletes standard tape volume labels, checks for unexpired files, and displays existing volume and data file labels.
Tape Error Summary	$TVES	Prints tape error statistics that have been accumulated during processing.
VTOC Service	$WVTOC	Removes the gaps between entries in a VTOC.

The information for most programs is divided into six sections:

- Program description
- Control statement summary
- Parameter summary
- Parameter descriptions
- OCL (operation control language) considerations
- Examples

PROGRAMMING CONSIDERATIONS

For information and reference, the following is a list of SCP system service programs and the minimum partition sizes required for execution:

$$RSTR	Checkpoint Restart	8K
$ALT	Alternate Track Assignment	8K
$BUILD	Alternate Track Rebuild	8K
$CNFIG	Configuration Record	8K
$COPY	Copy/Dump	10K
$DCOPY	Dump/Restore	8K
$DELET	File Delete	8K
$FCOMP	File Compress	10K
$HIST	System History Area Display	8K
$INIT	Disk Initialization	8K
$KLEAN	Chain Cleaning	8K
$LABEL	File and Volume Label Display	
	Simulation area VTOC	
	1000-file VTOC	8K
	1-1000 entries unsorted	10K
	1-300 entries sorted	10K
	301-500 entries sorted	12K
	501-700 entries sorted	14K
	701-900 entries sorted	16K
	901-1000 entries sorted	18K
$MAINT	Library Maintenance	10K
$MPXDV	Macro Processor	12K
$OLINK	Overlay Linkage Editor	10K
$QCOPY	Spool File Copy	10K
$RINDX	Recover Index	10K
$RSALT	Reassign Alternate Track	8K
$SCOPY	Simulation Area	10K
$TINIT	Tape Initialization	8K
$TVES	Tape Error Summary	8K
$WVTOC	VTOC Service	8K

CONTROL STATEMENTS

All system service programs except $TVES, $RINDX, and $KLEAN require that you supply control statements. These statements give the program information concerning the output you want the program to produce or the way in which you want the program to perform its function. The programs read these statements from the system input device. They must be the first input read by the program.

Every control statement is made up of an *identifier* and *parameters.* The *identifier* is a word that identifies the control statement. It is always the first word of the statement. Parameters are information you are supplying to the program. Every parameter consists of a keyword, which identifies the parameter, followed by the information you are supplying.

WRITING CONTROL STATEMENTS FOR SYSTEM SERVICE PROGRAMS

To write control statements, use the sections in the following ways:

1. Look at the *Control Statement Summary* to determine which control statements and parameters apply to the program functions you are interested in. (The program function(s) is stated in the text preceding the *Control Statement Summary*.)

2. If you need information about the contents or meanings of particuiar parameters, look at the *Parameter Summary.*

3. If you need more detailed information about parameters, read the *Parameter Descriptions* following the *Parameter Summary*.

4. If you need examples of specific jobs, look at the *Example* section. All examples show the OCL statements and control statements for specific jobs.

5. To find information concerning the use of the system service programs, refer to *OCL Considerations* for the necessary OCL statements.

Coding Rules

The rules for coding control statements are as follows:

1. *Statement identifier.* // followed by a blank should precede the statement identifier. Do not use blanks within the identifier.

2. *Blanks.* Use one or more blanks between the identifier and the first parameter. Do not use them anywhere else in the statement.

3. *Statement Parameters.* Parameters can be in any order. Use a comma to separate one parameter from another. Use a hyphen (-) within each parameter to separate the keyword from the information you supply. Do not use blanks within or between parameters.

4. *Statement parameters containing a list of data after the keyword.* Use apostrophes (') to enclose the items in the list. Use a comma to separate one item from another. Example: UNIT-'R1,R2' (R1 and R2 are the items in the list).

5. *Statement length.* No control statements, except for some disk initialization, library maintenance, and spool file copy statements may exceed 80 to 96 characters. The following library maintenance statements can be continued on another statement (see *Continuation* under *Coding Rules* in Part 1 of this manual):

   // ALLOCATE
   // COPY (not ENTRY statements or COPY statements read from a file)
   // DELETE
   // MODIFY (not REMOVE, REPLACE, or INSERT statements)
   // RENAME

   The disk initialization statement // VOL and the spool file copy statements // COPYRDRQ, // COPYPRTQ, // COPYPCHQ, // CHAIN, // RESTORE, // COPYQ, and // DISPLAY can also be continued.

The following is an example of a control statement:

```
// COPY FROM-F1,LIBRARY-O,NAME-SYSTEM,TO-R1
```

The statement identifier is COPY. The parameter keywords are FROM, LIBRARY, NAME, and TO. The information you supply is F1, O, SYSTEM, and R1.

END Control Statement

The END statement is a special control statement that indicates the end of control statements. It consists of // END starting in position 1 and must always be the last control statement for the programs.

Placement of Control Statements in the Job Stream

Control statements for system service programs must follow the RUN statement. The following example shows the use and placement of control statements.

```
OCL          { // LOAD $COPY,F1
statements   { // RUN
               // COPYPACK          }
                 FROM-F1,TO-R1     }  Control Statements
               // END               }  for the $COPY
                                    }  program
```

Special Meaning of Capital Letters, Numbers, and Special Characters

Capitalized words and letters, numbers, and special characters have special meanings in control statement descriptions.

In control statements, capitalized words and letters must be written as they appear in the statement description. Sometimes numbers appear with the capitalized information. These numbers must also be written as shown.

Words or letters that are not capitalized mean you must use a value that applies to the job you are doing. The values that can be used are listed in the parameter summaries for the control statements.

Braces { } and brackets [ ] sometimes appear in parameters shown in statement summaries and parameter summaries. They are not part of the parameter; they simply indicate a choice of values to complete the parameter. You *must* choose one of the values surrounded by braces; you *may* choose a parameter surrounded by brackets or omit that parameter entirely. Underscoring of one value enclosed by braces indicates the default. If you specify the keyword of a parameter, you must complete the parameter by supplying the code or data even though a default is indicated.

For example:

- [RECL-{80 / 96}] means that if you do not specify this parameter, the system will select RECL-96. If you specify the keyword RECL, you must also supply one of the values (80 or 96).
- RETAIN-{T / P} means that you must specify either RETAIN-T or RETAIN-P.
- [,BLKL-block length] means that the block length parameter may be omitted entirely.

DEVICE CODES

System service programs use parameters that require a code to indicate the unit being referenced. A unit code can refer to a main data area, simulation area, tape drive, or other device, depending on the function being performed.

Figure 4-1 shows the codes used for the main data areas and simulation areas. The simulation area codes are used in the ASSIGN statement, configuration record program ($CNFIG), and in the simulation area program ($SCOPY). In all other programs, a 5444 unit code (F1, R1, F2, R2) is used instead of the simulation area code. Before using these codes, you should be familiar with the format of each volume and the assignment of the codes as described in Part 3 of this manual.

Disk Drive Type and Designation	Main Data Area Codes	Simulation Area Codes
3340 drive 1	D1	D1A, D1B, D1C, D1D
3340 drive 2	D2	D2A, D2B, D2C, D2D
3340 drive 3	D3 or D31	D3E, D3A
3340 drive 4	D4 or D41	D4E, D4A
3344 drive 3		
Volume 1	D3 or D31	D3E, D3A
Volume 2	D32	D3F, D3B
Volume 3	D33	D3G, D3C
Volume 4	D34	D3H, D3D
3344 drive 4		
Volume 1	D4 or D41	D4E, D4A
Volume 2	D42	D4F, D4B
Volume 3	D43	D4G, D4C
Volume 4	D44	D4H, D4D

Figure 4-1. Main Data Area and Simulation Area Codes

Alternate Track Assignment Program—$ALT

PROGRAM DESCRIPTION

The alternate track assignment program has the following function:

Assign alternate tracks to disk tracks that become defective after they are initialized.

When the program assigns an alternate, it transfers the contents of the defective track to the alternate. Alternate tracks can replace any primary tracks except cylinder 0, head 0, on D1, D2, D3 or D31, D4 or D41. The cylinder 0, head 0, track is reserved for system use.

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

Function	Control Statements ①
Conditional Assignment	// ALT ② PACK-name,UNIT-code,VERIFY-number // END
Unconditional Assignment	// ALT ② PACK-name,UNIT-code,VERIFY-number,ASSIGN-track // END

① For each use, the program requires the statements in the order they are listed: ALT,END.

② There can be ten ALT statements per job.

PARAMETER SUMMARY

Parameter	Description
PACK-name	Name of the main data area.
UNIT-code	Location of the disk. Possible codes are those for the main data areas.
VERIFY-number	In testing the condition of a track, do surface analysis the number of times indicated (number can be 1-255). If VERIFY parameter is omitted, do surface analysis 16 times. The VERIFY parameter applies only to tracks in the suspected defective track list.
ASSIGN-track	The specified track is unconditionally assigned an alternate. Only one track can be assigned for each ALT statement. Valid track numbers are: 1-4187 for each volume on a 3340; 1-4199 for volumes D3 or D31 and D4 or D41 on a 3344; 0-4199 for volumes D32, D33, D34, D42, D43, D44 on a 3344.

PARAMETER DESCRIPTIONS

PACK Parameter

The PACK parameter (PACK-name) tells the program the name of the area to be processed. This is the volume label written by the disk initialization program. (See *Disk Initialization Program*.)

The alternate track assignment program compares the name in the PACK parameter with the volume label to ensure that they match.

UNIT Parameter

The UNIT parameter (UNIT-code) indicates the location of the area containing suspected defective tracks. Codes for the possible locations are those for the main data areas.

VERIFY Parameter

The VERIFY parameter (VERIFY-number) enables you to indicate the number of times you want the program to do surface analysis before judging whether or not the track is defective. The number can be from 1 to 255. If you omit the parameter, the program does surface analysis 16 times. This parameter applies only to tracks in the suspected defective track list.

ASSIGN Parameter

The ASSIGN parameter (ASSIGN-track) provides unconditional assignment of the specified track to the next available alternate track.

You can assign alternates for all tracks except 0 when the volume is on a 3340 (D1, D2, D3 or D31, D4 or D41). Valid track numbers are 1 through 4187.

You can assign alternates for all tracks except 0 on volumes D3 or D31 and D4 or D41 on a 3344. Valid track numbers are 1 through 4199.

You can assign alternates for all tracks on volumes D32, D33, D42, D43, and D44. Valid track numbers are 0 through 4199.

You can assign an alternate for only one track for each ALT statement you supply.

UNCONDITIONAL ASSIGNMENT

Unconditional assignment applies to tracks that occasionally cause read or write errors. Such tracks might not cause errors when tested by the alternate track assignment program ($ALT) during conditional assignment. If they do not, the program will not assign alternate tracks to them. If you still want to assign alternates to these tracks, use unconditional assignment. In doing unconditional assignment, the program assigns an alternate without first testing the condition of the track(s) suspected of being defective.

After a track has an assigned alternate, the only way to retest the track is to execute the disk initialization program ($INIT) with parameters TYPE-FORCE and ERASE-YES.

CONDITIONAL ASSIGNMENT

Conditional assignment consists of testing the condition of a track (surface analysis) and, if the track is defective, assigning an alternate track to replace it.

Situation: Conditional assignment applies to tracks that cause reading or writing errors during a job. Any time a track causes such errors, the system does the following:

1. Stops the program currently in operation
2. Writes the track address in the main data area defective track table
3. Halts with a halt code indicating a permanent disk I/O error

You can then run the alternate track assignment program.

When you use the alternate track assignment program to do conditional assignment, the program locates the tracks by using the addresses from the defective track table. The program will do conditional assignment for all identified tracks (one at a time) as long as there are alternate tracks available for assignment.

Surface Analysis: Surface analysis is a procedure the program uses to test the conditon of tracks. It consists of writing test data on a track, then reading the data to ensure that it was written properly.

Before doing surface analysis, the alternate track assignment program transfers any data from the track to an alternate track. This alternate is assigned if the track proves to be defective.

In judging whether or not the track is defective, the program does surface analysis the number of times you specify in the VERIFY parameter. If you omit the parameter, the program does surface analysis 16 times. If the track causes reading or writing errors any time during surface analysis, the program considers the track defective.

Assignment of Alternate Tracks: If a track proves to be defective, the program assigns an alternate track. The alternate becomes, in effect, a substitute for the defective track. Any time a program attempts to use the defective track, it automatically uses the alternate instead.

Each volume (a main data area and its associated simulation areas) has 40 alternate tracks. The program does not do conditional assignment if all alternate tracks are in use.

Note: $ALT does not process suspected defective tracks in any area that is currently being used. A message is used if suspected defective tracks are not processed.

Incorrect Data: If a track is defective, some of the data transferred to the alternate track could be incorrect. Therefore, when reading data from the defective track, the program prints all track records containing data that caused reading errors. Characters that have no print symbol are printed as two-digit hexadecimal numbers.

The following is an example:

```
ABCDE GH123 56 . . .
      B     A
      6     4
```

Appendix A lists the characters in the standard character set and their corresponding hexadecimal numbers.

To correct errors on the alternate track, use the alternate track rebuild program.

OCL CONSIDERATIONS

The following OCL statements are needed to load the alternate track assignment program:

```
// LOAD $ALT,code
// RUN
```

The code you supply depends on the location of the simulation area containing the alternate track assignment program. Possible codes are R1, F1, R2 and F2.

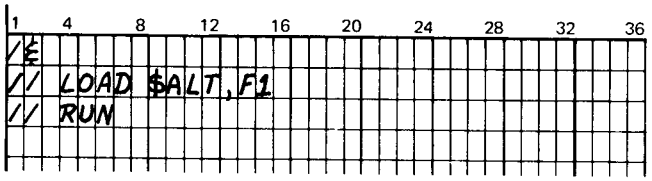
EXAMPLES

Conditional Assignment

Figures 4-2 and 4-3 are examples of the OCL statements and control statements needed for a conditional assignment as described in the following situation.

Situation

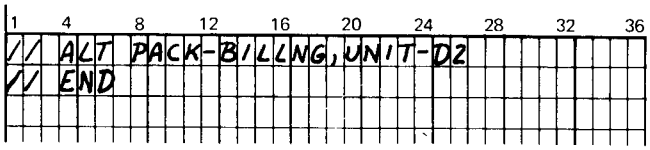
The system cancels a job if a defective track is found on the main data area on drive 2. (The name of the disk is BILLNG.) Before doing more jobs, the operator wants to use the alternate track assignment program to check the condition of the track and assign an alternate to the track if it is defective.



Explanation:

Alternate track assignment program is loaded from the F1.

Figure 4-2. OCL Load Sequence for Alternate Track Assignment



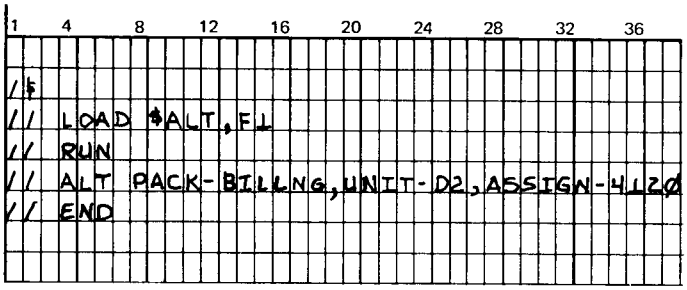
Explanation:

- The name of the disk (BILLNG) and its location (main data area on drive 2) are indicated by the PACK and UNIT parameters in the ALT statement.
- Because the VERIFY parameter was omitted from the ALT statement, the program does surface analysis 16 times when it tests the condition of suspected defective tracks.

Figure 4-3. Control Statements for a Conditional Assignment

Unconditional Assignment

Figure 4-4 is an example of the OCL statements and control statements needed for an unconditional assignment.



Explanation:

- The name of the disk (BILLNG) and its location (main data area on drive 2) are indicated by the PACK and UNIT parameters in the ALT statement.
- Because of the ASSIGN parameter, the program assigns track number 4120 an alternate without testing its condition.

Figure 4-4. OCL and Control Statements for an Unconditional Assignment

MESSAGES FOR ALTERNATE TRACK ASSIGNMENT

Message	Meaning
ALTERNATE TRACK ASSIGNED	This message is printed when an alternate track has been assigned to a defective track and the data has been transferred to the alternate track.
PRIMARY TRACK HAS BEEN TESTED OK TRACK xxxx,UNIT-zzz	This message is printed when the program determines that a primary track is not defective. xxxx is the primary track number and zzz is the volume involved.
RECORD WITH DATA ERROR	This message is printed when the alternate track assignment program found an error when transferring data. The record that has the error is printed out.
PRIMARY TRACK xxxx ALTERNATE TRACK yyyy,UNIT-zzz	This message is printed after ALTERNATE TRACK ASSIGNED. xxxx is the primary track number, yyyy is the alternate track number, and zzz is the volume involved.

Alternate Track Rebuild Program—$BUILD

PROGRAM DESCRIPTION

The alternate track rebuild program has the following function:

Enable you to correct data that could not be transferred correctly to an alternate track.

One or more alternate tracks can be corrected by the system during one execution of this program. You must supply the data to correct the errors.

To write control statements for this program, you need the information printed by the alternate track assignment program when it assigned the alternate track. The printed information tells you the name of the area and numbers of the track and records suspected of containing incorrect data. It also includes the data from these records that you can use to locate incorrect data. Fixed record refers to a physical 256-byte record.

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

Function	Control Statements and Substitute Data

To replace characters 1-12 and 75-78 of a sector, you can use either of the following:
• Use one REBUILD statement and replace all the characters specifying a LENGTH parameter of 78.
• Use one REBUILD statement for every set of positions you correct.
The data you want to substitute must follow the REBUILD statements to which it applies. The order of the control statements and data in the preceding example would be:
// REBUILD statement
data for positions 1-78
// END
 	
// REBUILD statement for positions 1-12
data
// REBUILD statement for positions 75-78
data
// END
	// REBUILD PACK-name,UNIT-code,TRACK-location,LENGTH-number,DISP-position
	Substitute data
	// END

PARAMETER AND SUBSTITUTE DATA SUMMARY

REBUILD Statement

Parameter	Description
PACK-name	Name of the area.
UNIT-code	Location of the disk. Possible codes are those for the main data areas.
TRACK-location	Number of track and fixed record containing incorrect data. Number is printed by alternate track assignment program. Track number must be four digits; fixed record number must be two digits. (TRACK-011109 means track 111, fixed record 9.)
LENGTH-number	Number of characters being replaced. Number can be 2-256 and must be a multiple of 2 (2, 4, 6, etc).
DISP-position	Position of the first character being replaced in the record. Position can be 1-255.

Substitute Data: Code each character in hexadecimal form. Follow every second character, except the last, with a comma. Example: The numbers 123456 would be coded as F1F2,F3F4,F5F6. (Appendix A lists the hexadecimal codes for System/3 characters.)

PARAMETER AND SUBSTITUTE DATA DESCRIPTIONS

PACK Parameter

The PACK parameter (PACK-name) tells the program the name of the area that contains the alternate track being corrected. This is the volume label written by the disk initialization program.

The alternate track rebuild program compares the name in the PACK parameter with the volume label to see if they match.

UNIT Parameter

The UNIT parameter (UNIT-code) indicates the location of the area that contains the alternate track being corrected. Possible codes are those for the main data areas.

TRACK Parameter

The TRACK parameter (TRACK-location) identifies the track and record containing the data being corrected. The defective track, not the alternate track, is the one you refer to. Referencing the defective track is the same as referencing the alternate track.

For the main data area, the possible track numbers are 0001-4184. Always use four digits. The possible fixed record numbers are 01-48. Always use two digits. The track number must precede the fixed record number. For example, the parameter TRACK-111019 means track 1110, record 19.

Track and record numbers are printed by the alternate track assignment program when it prints data from records that contain incorrect data.

LENGTH Parameter

The LENGTH parameter (LENGTH-number) tells the program how many characters you are replacing in the fixed record. You must replace characters in multiples of 2 (2, 4, 6, and so on). The maximum is 256, which is the capacity of a fixed record.

Length applies to characters that occupy consecutive positions in the fixed record. If the characters you want to replace do not occupy consecutive positions, you must either replace all intervening characters or use more than one REBUILD statement. For example, to replace characters 10-11 and 24-25 in a fixed record, you can do either of the following:

- Use one REBUILD statement to replace characters 10-25 (LENGTH-16).
- Use two REBUILD statements to replace characters 10-11 (LENGTH-2) and 24-25 (LENGTH-2).

DISP (Displacement) Parameter

The DISP parameter (DISP-position) indicates the position of the first character being replaced in the fixed record. The position of the first character is 1; the position of the second character is 2, and so on. The maximum position you can specify is 255.

Beginning at the position you indicate, the alternate track rebuild program replaces the number of characters you indicate in the LENGTH parameter.

Substitute Data

After each REBUILD statement, you must code the substitute characters that apply to that statement. The characters must be in hexadecimal form. Appendix A shows the hexadecimal codes for the System/3 character set.

Include a comma after every second character. For example, the data F1F2,F3F4,F5F6 represents 123456. F1 is the hexadecimal form of 1; F2 is the hexadecimal form of 2, and so on.

Code only the number of characters you indicate in the LENGTH parameter in the REBUILD statement.

Note: If the LENGTH parameter of the REBUILD statement exceeds 38, at least two substitute data statements are required. Each substitute data statement, except the last one, must be completely filled with data and must have a comma in column 95 and a blank in column 96. For an 80 column input device, it is possible to have only one substitute data statement.

OCL CONSIDERATIONS

The following OCL statements are needed to load the alternate track rebuild program.

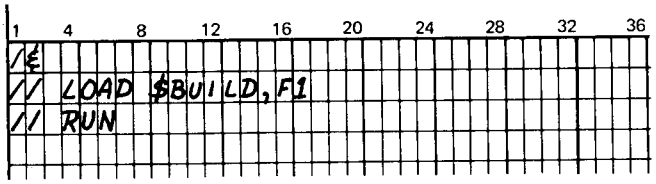
```
// LOAD $BUILD,code
// RUN
```

The code you supply depends on the location of the simulation area containing the alternate track rebuild program. Possible codes are R1, F1, R2, F2.

EXAMPLES

Correcting Characters on an Alternate Track

Figures 4-5 and 4-6 are examples of the OCL and control statements needed for correcting characters on an alternate track.



Explanation:

Alternate track rebuild program is loaded from F1.

Figure 4-5. OCL Load Sequence for Alternate Track Rebuild

Situation

Assume that the alternate track assignment program ($ALT) printed the following information:

```
**RECORD WITH DATA ERROR**
TRACK   1.......10........20........30........40........50........60........70........80......88
020001
        00000000000000001111111111111111222222222222222233333333333333333 444444444444455555555
        0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF 123456789ABCDEF01234567

        55555555666666666666666677777777777777778888888888888888999999999999999AAAAAAAAAAAAAAAA
        89ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
                      ABCDEFGHI       JKLMNOPQR        STUVWXYZ      0123456789
        BBBBBBBBBBBBBBBC          CCCCCCD          DDDDDDEF         EEEEEE           FFFFFF
        0123456789ABCDEF0         ABCDEF0          ABCDEF01         ABCDEF           ABCDEF
```

It means that errors were detected in record 1 of track 200. (Assume the name of the area is BILLNG.)

In checking the characters printed by the alternate track assignment program, you found that the characters in positions 120-123 in the record are incorrect, and you want the operator to run the alternate track rebuild program to correct them.

```
1   4   8   12  16  20  24  28  32  36  40  44  48  52  56  60
// REBUILD PACK-BILLNG,UNIT-D2,TRACK-020001,LENGTH-4,DISP-120
C7C8,C9F1
// END
```

Explanation:

- The name of the main data area (BILLNG) and its location (drive 2) are indicated in the PACK and UNIT parameters in the REBUILD statement.
- The record containing the incorrect characters is record 1 of the alternate track assigned to track 200 (TRACK-020001). The character in position 120 is the first character being replaced (DISP-120).
- The characters in positions 120 through 123 in record 1 are being replaced (LENGTH-4).
- The substitute characters follow the REBUILD statement. They are G (C7), H (C8), I (C9), and I (F1).

Figure 4-6. Control Statements for Correcting Characters on an Alternate Track

Configuration Record Program—$CNFIG

PROGRAM DESCRIPTION

The configuration record program has the following functions:

- Change the assignment of the 5444 unit codes.
- Set the automatic start for spooling functions on or off.
- Set the automatic write for spooling functions on or off.
- Allow or disallow the blank OCL function.
- Change the support for cataloging to a program pack.
- Allow or disallow sharing among partitions of the console as a system input device.
- Change the spooling card type, forms type, or track group size.
- Change the system date format.
- Change the halt status for a partition.
- Change the halt status for the system history area.
- Change the system IDELETE status.
- Change the log device and eject status for a partition.
- Delete or retain I-type messages that are on the console at EJ.
- Format and print the configuration record.
- Change the priority of partitions and spool during IPL.
- Allow or disallow $QCOPY to erase the display screen before writing the next message to the terminal. ($QCOPY must be executing under CCP.)
- Change the requirement for $QCOPY authorization.
- Change the limit of the priority of jobs placed on the active spool file reader queue by $QCOPY. ($QCOPY must be executing under CCP.)
- Allow messages to be displayed while ENTER READER DATA or ENTER DATA is prompted on the console.
- Change the extended restart function.
- Change the warning point for SHA overlay.
- Change the size of main storage, selected partitions, and file share area.
- Change the spool file cylinder size.
- Change the spool file location.
- Include the page/card count and the form/card type information on the spool time messages that are entered in the system history area.
- Change the system input, punch, or print device assigned to a partition.
- Allow or disallow time stamping of I-, D-, and R-type messages that appear on the system console.
- Change the ratio of share DTFs and FSQEs in the file share area.
- Change the file share default option.

Note: Before executing this program, you should be familiar with the information about simulation areas in Part 3 of this manual and also the information presented under *Considerations for Program Pack Protection* in the *IBM System/3 Model 15 System Generation Reference Manual,* GC21-7616.

Changing the Configuration Record

The parameter that you specify on the control statement changes the configuration record on the system pack from which the IPL was performed. Therefore, it is not necessary to do a system generation for the purpose of changing any part of the configuration record that can be changed by the $CNFIG program.

Although the configuration record on the system pack is changed at the time $CNFIG is executed, the change is not effective until another IPL is performed. The change to the configuration record remains in effect until another system generation or until it is changed by $CNFIG.

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

Function	Control Statements
Assign 5444 unit codes to partition 1, 2, or 3 simulation areas	// ASNP {1 / 2 / 3} code-area
Set automatic start for spooling functions on or off	① // AUTST [PRINT- {YES / NO}] [,PUNCH- {YES / NO}] [,READ- {YES / NO}]
Set automatic write for spooling functions on or off	① // AUTWT [PRINT- {YES / NO}] [,PUNCH- {YES / NO}]
Blank OCL	// BLANK OCL- {YES / NO}
Catalog to a program pack	// CATLG PACK- {NONE / CCP / ALL}
Specify whether the system console can be shared by the partitions as a system input device	// CONSOL SHARE- {YES / NO}
Change spooling card type	// DEFCN CARD-xxx
Change spooling forms type	// DEFFN FORM-xxx
Change system date format	// FORMAT DATE- {MMDDYY / DDMMYY}
Change ratio of share DTFs to FSQEs and/or file share default option	① // FSHARE [RATIO- {1 / 2 / 3 / 4}] [,SHARE- {YES / NO}]
Halt partition 1, 2, or 3 on system messages	// {HLTP1 / HLTP2 / HLTP3} HALT- {YES / NO}
Set the delete I-messages function	// ITYPE IDELETE- {YES / NO}

① You must specify at least one parameter.

Function	Control Statements
Reassign partition 1, 2, or 3 log device and set eject status	// {LOGP1 / LOGP2 / LOGP3} DEVICE-{3277 / 1403 {,EJECT / ,NOEJECT} / 3284 {,EJECT / ,NOEJECT}}
Specify whether I-type messages on the screen at EJ are to be retained	// MESSAG RETAIN-{YES / NO}
Print configuration record	// PRINT
Specify the priority of partitions and spool during IPL	// PRIORITY SEQUENCE-'{SPOOL / (SP) / P1 / P2 / P3} {[= / , / -]} . . . '
Specify whether $QCOPY is to erase the screen before writing the next message; change the authorization requirement for $QCOPY; change the limit of the priority of jobs placed on the active spool file reader queue by $QCOPY	(1) // QCOPY [(2) ERASE-{YES / NO}] [,AUTHORIZE-{YES / NO}] [(2) ,RQPTY-{0 / 1 / 2 / 3 / 4 / 5}]
Set extended restart function on or off	// READY [EXTENDED-{YES / NO}] [,MESSAGE-{YES / NO}]
Halt system or invoke $HACCP when the SHA is xx tracks from overlaying unprinted records	(1) // SHA [HALT-{YES / NO / CCPAUTO}] [,TRACKS-xx]
Change the size of main storage, partitions 1, 2, and 3, and/or the file share area	(1) // SIZE [SYS-xxx] [,P1-xxx] [,P2-xxx] [,P3-xxx] [,FS-xxx]
Change the spool file cylinder size	// SPCYL CYL-xxx
Change the location of the spool file	// SPDSK UNIT-code

(1) You must specify at least one parameter.
(2) This parameter does not perform any function unless $QCOPY is being executed under control of the communications control program (CCP).

Function	Control Statements
Change spooling track group size	// SPEXT TRACKS-{1 / 2 / 4 / 5 / 10}
Include page/card count and form/card type information on the spool time message	// SPOPT M-{YES / NO}
Reassign partition 1, 2, or 3 system input device	// SYIN{1 / 2 / 3} DEVICE-{MFCU1 / MFCU2 / MFCM1 / MFCM2 / 1442 / 2501 / CONSOLE / 3741}
Reassign partition 1, 2, or 3 system punch device	// SYPC{1 / 2 / 3} DEVICE-{MFCU1 / MFCU2 / MFCM1 / MFCM2 / 1442 / 3741 / NONE}
Reassign partition 1, 2, or 3 system print device	// SYPR{1 / 2 / 3} DEVICE-{1403 / 3284}
Time stamp I-messages, or D- and R-messages and their responses	// TSTAMP I-{NO / YES}, D-{NO / YES}

This page intentionally left blank.

PARAMETER SUMMARY

Parameter	Description
AUTHORIZE-YES -NO	Specifies whether authorization is required for $QCOPY.
CARD-xxx	xxx can be a maximum of three valid IBM System/3 characters except commas, leading or embedded blanks, or apostrophes.
code-area	Possible 5444 unit codes are F1, R1, F2, R2 Possible simulation area codes are D1A, D1B, D1C, D1D, D2A, D2B, D2C, D2D, D3A, D3B, D3C, D3D, D3E, D3F, D3G, D3H, D4A, D4B, D4C, D4D, D4E, D4F, D4G, D4H
CYL-xxx	Specifies the new total number of cylinders for the spool file. The number (xxx) must be from 1 through 166 (3340) or 1 through 186 (3344).
DATE-MMDDYY -DDMMYY	Month-day-year Day-month-year
D-NO -YES	If D-YES is specified, D-(decision) and R-(reply) type messages that appear on the system console and their responses are time-stamped when entered in the SHA (system history area). If D-NO is specified or assumed, D-type messages are not time-stamped.
DEVICE-MFCM1	Primary hopper of 2560
-MFCM2	Secondary hopper of 2560
-MFCU1	Primary hopper of 5424
-MFCU2	Secondary hopper of 5424
-1403	1403 Printer
-1442	1442 Card Read Punch
-2501	2501 Card Reader
-CONSOLE	3277 Display Station
-3284	3284 or 3287 Printer
-3741	Directly attached 3741 Data Station/Programmable Work Station
-NONE	No punch device assigned
EJECT/NOEJECT	Specifies whether a page is to be ejected before ES and EJ and after EJ.

Parameter	Description
ERASE-YES -NO	Specifies whether $QCOPY (when executed under CCP) is to erase the terminal screen after information has been entered on the terminal.
EXTENDED-YES -NO	Specifies whether the extended restart function is to be used.
FORM-xxx	xxx can be a maximum of three valid IBM System/3 characters except commas, leading or embedded blanks, or apostrophes.
FS-xxx	Specifies the new size (a multiple of 2) of the file share area. The number (xxx) cannot be less than 2.
HALT-YES -NO	Specifies whether the system is to halt, not halt, or invoke $HACCP when a predetermined number of tracks remain in the SHA before unprinted entries are overlaid.
I-NO -YES	If I-YES is specified, I-type (informational) messages that appear on the system console are time-stamped when entered in the SHA (system history area). If I-NO is specified or assumed, I-type messages are not time-stamped.
IDELETE-YES -NO	Specifies whether I-type messages are to be deleted.
M-YES -NO	Specifies whether to include the page/card count and the form/card type information on the spool message in the system history area.
OCL-YES -NO	Specifies whether the blank OCL function is supported.
PACK-NONE -CCP -ALL	No cataloging to any program pack. Cataloging to the CCP program pack. Cataloging to all program packs.
PRINT-YES -NO	Specifies whether the print writer is to automatically start during IPL (AUTST). Or, specifies whether the print writer is to automatically start when output is available on the print queue (AUTWT).
PUNCH-YES -NO	Specifies whether the punch writer is to automatically start during IPL (AUTST). Or, specifies whether the punch writer is to automatically start when output is available on the punch queue (AUTWT).
P1-xxx	Specifies the new size (a multiple of 2) of partition 1. The minimum size of P1 is 8K.
P2-xxx	Specifies the new size (a multiple of 2) of partition 2. The minimum size of P2 (if other than zero) is 8K.

Parameter	Description
P3-xxx	Specifies the new size (a multiple of 2) of partition 3. The minimum size of P3 (if other than zero) is 8K.
RATIO-1 -2 -3 -4	Specifies the ratio of share DTFs to FSQEs in the share area.
READ-YES -NO	Specifies whether the spooled reader is to automatically start after an IPL is performed.
RETAIN-YES -NO	Specifies whether to retain the I-type messages that are on the console at EJ.
RQPTY-0 -1 -2 -3 -4 -5	Specifies the priority limit for the jobs that are placed on the active spool file reader queue by $QCOPY. This function is effective only when $QCOPY is executed under control of CCP.
SEQUENCE-'(P1 / P2 / P3 / SPOOL / SP){[=]}...'	Specifies the priority of partitions and spool.
SHARE-YES -NO	Specifies whether file sharing is allowed between partitions (FSHARE). Or, specifies whether the console can be shared among partitions as an input device (CONSOL).
SYS-xxx	Specifies the main storage capacity of the system. Possible values for xxx are 96, 128, 160, 192, 224, 256, 384, or 512.
TRACKS-1 -2 -4 -5 -10	The size of track groups within the spooling disk area.
TRACKS-xx	Specifies the number of tracks in the SHA between the SHA warning point and the first unprinted entry. Possible values for xx are 0 through 10.
UNIT-code	Specifies the main data area that is to contain the spool file. Possible codes are those for the main data areas.

This page intentionally left blank.

PARAMETER DESCRIPTIONS

AUTHORIZE Parameter (QCOPY)

The AUTHORIZE parameter is associated with the authorization function of the Spool File Copy program ($QCOPY). When AUTHORIZE-YES is specified and $QCOPY is executed under CCP, certain functions of $QCOPY cannot be executed unless the user is properly authorized. When AUTHORIZE-NO is specified, no authorization is required. For more information about granting authorization to a user, refer to the *Spool File Copy program, $QCOPY.*

CARD Parameter (DEFCN)

The CARD parameter (CARD-xxx) redefines the card type that the operator loads into the punch device for the next job. *xxx* can be any three valid System/3 characters (see Appendix A) except commas or leading or embedded blanks. The CARDNO parameter of a PUNCH OCL statement can temporarily override this definition.

Code-Area Parameter (ASNPx)

Simulation area codes are unique to the type and designation of each disk drive. After assignment, the 5444 unit code is used to identify the simulation area to be accessed. The following chart shows the relationship of the type and designation of a disk drive and the simulation area codes.

Disk Drive Type and Designation	Simulation Area Codes
3340 drive 1	D1A, D1B, D1C, D1D
3340 drive 2	D2A, D2B, D2C, D2D
3340 drive 3	D3E, D3A
3340 drive 4	D4E, D4A
3344 drive 3	
Volume 1	D3E, D3A
Volume 2	D3F, D3B
Volume 3	D3G, D3C
Volume 4	D3H, D3D
3344 drive 4	
Volume 1	D4E, D4A
Volume 2	D4F, D4B
Volume 3	D4G, D4C
Volume 4	D4H, D4D

Restrictions for Assigning Simulation Areas

Unit codes (F1, R1, F2, and R2) are assigned by partition to any of the supported simulation areas. Each unit code for a given partition must be assigned to a simulation area. Simulation area assignments are set in the configuration record during system generation and remain in effect until either another system generation is performed that has different simulation area assignments or until the configuration record program ($CNFIG) is used to change the simulation area assignments. An ASSIGN statement temporarily reassigns the simulation areas for the partition in which it is processed until another IPL is performed or another ASSIGN statement is processed.

Refer to the following chart for invalid simulation area assignments.

5444 Unit Codes	Simulation areas that cannot be assigned the 5444 unit codes.
F1	D1B, D3B
R1	D1A, D3A
F2	D1A, D1B, D3A, D3B
R2	D1A, D1B, D3A, D3B

CYL Parameter (SPCYL)

The CYL parameter (CYL-xxx) specifies the new total number of cylinders assigned to the spool file. The number (xxx) must be from 1 through 166 if the spool file is on a 3340 Direct Access Storage Facility or from 1 through 186 if the spool file is on a 3344 Direct Access Storage.

D Parameter (TSTAMP)

The D-YES parameter specifies that decision and reply type messages that appear on the system console and their responses are to be time-stamped when entered in the system history area. If D-NO is specified or assumed, the messages and their responses are not time-stamped.

The time-stamp record immediately follows the message and response. The following example shows the format of the time-stamp record:

```
2 // LOAD $MAINT,F1
2 // RUN
2 // COPY F
2 - 2 LM 6A ST D  123              $MAINT01
2     04/26/78 00.00.59
2 2 2 LM 6A ST D  123              $MAINT01
2     04/26/78 00.01.02
```

DATE Parameter (FORMAT)

The DATE parameter redefines the system date format to either month-day-year (DATE-MMDDYY) or day-month-year (DATE-DDMMYY).

DEVICE Parameter (LOGPx, SYINx, SYPCx, SYPRx)

The DEVICE parameter redefines a partition's assigned log device, system input device, system punch device, or system print device.

Valid codes for the partition log device are 3277, 1403, 3284. An EJECT or NOEJECT parameter can be specified with the 1403 or 3284. A LOG OCL statement can override this definition.

Valid codes for the system input device are MFCM1, MFCM2, MFCU1, MFCU2, 1442, 2501, 3277, 3741. A READER command or READER OCL statement can override this definition.

Valid codes for the system punch device are MFCM1, MFCM2, MFCU1, MFCU2, 1442, 3741, or NONE. A PUNCH OCL statement can override this definition.

Valid codes for the system print device are 1403 or 3284. A PRINTER OCL statement can override this definition.

EJECT or NOEJECT Parameter (LOGPx)

The EJECT or NOEJECT parameter, when included with the DEVICE-1403 or DEVICE-3284 parameter of the LOGPx statement, specifies whether a page is to be ejected before end of step, before end of job, and after end of job. A LOG OCL statement can override this specification.

ERASE Parameter (QCOPY)

When ERASE-YES is specified and $QCOPY is executed under CCP, the screen on the requesting terminal is erased before each message is written. When ERASE-NO is specified, the information entered on the terminal remains on the screen. The operator can erase the screen by pressing the ERASE EOF key.

EXTENDED Parameter (READY)

The EXTENDED parameter (EXTENDED-YES or EXTENDED-NO) specifies whether the automatic message restart function is to be used. This parameter applies only to the extended restart function and not to the unit record restart function. If used, the system operator need not respond with a 1 option to certain system messages.

Note: This parameter is valid only if the unit record restart and/or the extended restart option(s) was selected during system generation.

FORM Parameter (DEFFN)

The FORM parameter (FORM-xxx) redefines the forms type that the operator mounts on the printer for the next job. *xxx* can be any three System/3 characters (see Appendix A) except commas or leading or embedded blanks. The FORMSNO parameter of a PRINTER OCL statement or a CHANGE FRM command can temporarily override this definition.

FS Parameter (SIZE)

The FS parameter (FS-xxx) specifies the new size (in multiples of 2K) of the file share area. The size (xxx) cannot be less than 2 or greater than the size of main storage minus the sum of the amount of main storage assigned to the supervisor, partition 1, partition 2, and partition 3.

HALT Parameter (HLTPx)

The HALT parameter (HALT-YES or HALT-NO) specifies whether the system is to halt a partition for system messages (HLTPx). A HALT OCL statement or a HALT command can override this definition.

This HALT parameter functions as an OCL halt and cannot be overridden by an OCC NOHALT command.

HALT Parameter (SHA)

The HALT-YES parameter specifies that the system is to issue a message and stop when a predetermined number of tracks remain in the system history area before overlay of unprinted records occurs. The HALT-NO parameter causes no action to be taken when unprinted records in the system history area are about to be overlaid. The HALT-CCPAUTO parameter specifies that the system is to invoke the $HACCP program (if CCP is executing) when a predetermined number of tracks remain in the system history area before overlay of unprinted records occurs.

I Parameter (TSTAMP)

The I-YES parameter specifies that informational type messages that appear on the system console are to be time-stamped when entered in the system history area. If I-NO is specified or assumed, the messages are not time-stamped.

The time-stamp record immediately follows the message. The following example shows the format of the time-stamp record:

```
2 - 2 SP UT FA I
      SPOOL FILE STARTED - OLD,D43,50,5
2     05/08/78 00.00.17
```

IDELETE Parameter (ITYPE)

The IDELETE parameter (IDELETE-YES or IDELETE-NO) specifies whether the system is to automatically delete information-type messages that have not been responded to. Normally, a response is required to delete these messages. An IDELETE or NOIDELETE command can override this specification.

M Parameter (SPOPT)

When M-YES is specified, the page count and the forms type for print queue job steps are included on the spool time message in the system history area. Also, the card count and the card type for punch queue job steps are included on the spool time message in the system history area. When M-NO is specified, this additional information is not included on the spool time message.

The format of the messages is shown in the following examples.

```
S SP UT TM I JOBA          IODRVR01
  12/08/78 00.01.37 00.01.45  PR
  FORM TYPE-10A PAGES-2

S S PUT TM or I JOBB       IODRVR02
  12/08/78 00.01.46 00.01.59  PC
  CARD TYPE-XYZ CARDS-10
```

OCL Parameter (BLANK)

The OCL parameter (OCL-YES or OCL-NO) specifies whether the blank OCL function is to be used. If the function is used, all statements entered via the system input device (when that device is the system console) are erased from the screen when the ENTER key is pressed.

PACK Parameter (CATLG)

The PACK-NONE parameter prevents cataloging of an object library entry to any simulation area that contains a program which another partition has loaded and is currently executing.

The PACK-CCP parameter allows cataloging of an object library entry to a CCP program pack even though another partition (in addition to the CCP partition) has loaded and is currently executing a program from the CCP program pack.

The PACK-ALL parameter allows cataloging of object library entries to all program packs even though one or more partitions have loaded programs from them and are currently executing those programs.

For other restrictions, see *Library Maintenance Program* under *Multiprogramming Considerations and Restrictions* in Part 2 of this manual.

PRINT Parameter (AUTST, AUTWT)

The PRINT parameter (PRINT-YES or PRINT-NO), when used with AUTST, specifies whether the print writer automatically starts when an IPL is performed. This option relieves the operator of having to initially enter an operator control command to start the printer when spooling is specified.

The PRINT parameter, when used with AUTWT, specifies whether the print writer is to automatically start whenever output is available on the print queue. When output is not available on the queue, the print writer waits for output without issuing a message or requiring operator interaction.

PUNCH Parameter (AUTST, AUTWT)

The PUNCH parameter (PUNCH-YES or PUNCH-NO), when used with AUTST, specifies whether the punch writer is to automatically start when an IPL is performed. This option relieves the operator of having to initially enter an operator control command to start the punch when spooling is specified.

The PUNCH parameter, when used with AUTWT, specifies whether the punch writer is to automatically start whenever output is available on the punch queue. When output is not available on the queue, the punch writer waits for output without issuing a message or requiring operator interaction.

Note: Punch autostart (// AUTST PUNCH-YES) is valid only if reader autostart is not in effect.

P1 Parameter (SIZE)

The P1 parameter (P1-xxx) specifies the new size (a multiple of 2) of partition 1. The number (xxx) cannot be less than 8 or greater than the size of main storage minus the sum of the sizes of the supervisor, partition 2, partition 3, and the file share area.

P2 Parameter (SIZE)

The P2 parameter (P2-xxx) specifies the new size (a multiple of 2) of partition 2. The number (xxx) cannot be less than 8 (if other than zero) or greater than the size of main storage minus the sum of the sizes of the supervisor, partition 1, partition 3, and the file share area.

P3 Parameter (SIZE)

The P3 parameter (P3-xxx) specifies the new size (a multiple of 2) of partition 3. The number (xxx) cannot be less than 8 (if other than zero) or greater than the size of main storage minus the sum of the sizes of the supervisor, partition 1, partition 2, and the file share area.

RATIO Parameter (FSHARE)

When an IPL is performed or the file share area size is set via an OCC command, the share area is initialized according to the following specified ratio.

Ratio	Share Area Size (K bytes)	Number of FSQEs	Number of SDTF Blocks[1]
1	2	23	24
	4	35	52
	6	39	83
	8	49	108
2	2	23	24
	4	51	47
	6	71	73
	8	97	93
3	2	39	19
	4	67	42
	6	86	68
	8	113	88
4	2	39	19
	4	83	37
	6	119	58
	8	145	78

[1] Refer to *File Sharing* for a description of SDTFs and FSQEs.

Note: If the RATIO parameter is not specified, a ratio of 1 is assumed.

READ Parameter (AUTST)

The READ parameter (READ-YES or READ-NO) specifies whether the spool reader is automatically started when an IPL is performed. This option relieves the operator of having to initially enter certain operator control commands when spooling is specified. (Reader autostart (// AUTST READ-YES) is valid only if punch autostart is not in effect.)

RETAIN Parameter (MESSAG)

When RETAIN-NO is specified, I-type messages are erased from the system console screen at EJ. When RETAIN-YES is specified, I-type messages are not erased from the system console screen at EJ.

RQPTY Parameter (QCOPY)

The RQPTY parameter (RQPTY-0 through 5) specifies the priority limit of the jobs placed on the active spool file reader queue by $QCOPY. (This parameter is effective only when $QCOPY is executed under control of CCP.) The HIPTY OCC command can be used to temporarily override this parameter. For more information about reader queue priority, refer to the *Spool File Copy program ($QCOPY).*

SEQUENCE Parameter (PRIORITY)

The SEQUENCE parameter specifies the priority of the partitions and spool.

The order of the code parameters specifies the priority. The leftmost code parameter has the highest priority; the rightmost code parameter has the lowest. Two or more tasks have equal priority if an equal sign (=) or a hyphen (-) is specified between the parameters.

The following rules apply to the code parameters that can be supplied with the SEQUENCE keyword.

- Each code parameter can be specified only once.
- SPOOL (SP) is an invalid code parameter if spooling is not supported.
- All code parameters must be specified; there are no defaults.

The following examples illustrate these rules:

```
SEQUENCE-'P1,P2,P3,SP'
SEQUENCE-'P1,P2,P3'
SEQUENCE-'P1=P2=P3'
SEQUENCE-'P1-P2-P3'
SEQUENCE-'SP,P2,P1,P3'
SEQUENCE-'P1,P2=P3-SP'
```

- Only one group of code parameters can be equated. For example:

```
SEQUENCE-'SP=P3,P2=P1'
is not valid, but
SEQUENCE-'SP-P3-P2-P1'
and
SEQUENCE-'SP,P3=P2=P1'
are valid.
```

SHARE Parameter (CONSOL)

The SHARE parameter (SHARE-YES) specifies that the console can be shared by the partitions as a system input device. When SHARE-NO is specified, the console cannot be shared by the partitions.

SHARE Parameter (FSHARE)

The SHARE parameter (SHARE-YES or SHARE-NO) specifies whether the system will default to allow file sharing between partitions.

SYS Parameter (SIZE)

The SYS parameter (SYS-xxx) specifies the new main storage size for the system. Possible values for xxx are:

Group 1	Group 2
96	384
128	512
160	
192	
224	
256	

The configuration record program will issue an error message when a value is specified that is not within the same group as the current system size. This is because group 1 and group 2 require different modules to be included in the generated supervisor. Therefore, a system generation is required when it is necessary to change the system size to a value that is not within the same group as the current system size.

TRACKS Parameter (SHA)

The TRACKS parameter (TRACKS-0 through 10) defines the SHA warning point. The xx value specifies the number of tracks that remain in the SHA between the warning point and the first unprinted entry. When xx tracks remain and HALT-YES is specified on the SHA statement, the system stops and a message is displayed. When xx tracks remain and HALT-NO is specified on the SHA statement, the system is not stopped and no message is displayed. When xx tracks remain (xx must not be 0) and HALT-CCPAUTO is specified on the SHA statement, the $HACCP program is invoked (if CCP is executing). For information about the $HACCP program, refer to the *IBM System/3 Communications Control Program System Reference Manual,* GC21-7620.

The number xx must not be greater than one-half the number of tracks in the SHA.

If TRACKS-0 is specified, the system assumes a warning point of approximately 50 40-byte SHA records. (There are 144 40-byte SHA records per track.)

TRACKS Parameter (SPEXT)

The TRACKS parameter (TRACKS-nn) changes the size of the track groups within the spooling disk area. *nn* can be 1, 2, 4, 5, or 10. The START command can override this definition.

UNIT Parameter (SPDSK)

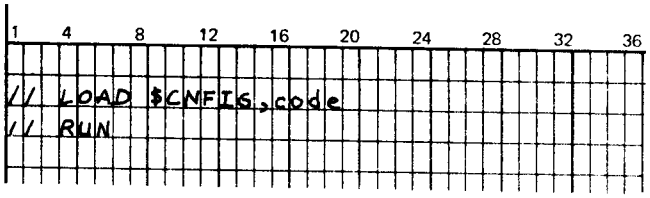
The UNIT parameter (UNIT-code) specifies the new main data area to be used for the spool file. Possible codes are D1, D2, D3, D31, D32, D33, D34, D4, D41, D42, D43, and D44.

CONSIDERATIONS AND RESTRICTIONS

- This program must be executed on a dedicated system except when printing the configuration record.
- The changes are not effective until an IPL is performed.
- The new configuration record remains in effect until another system generation or until it is changed by $CNFIG.
- The ASSIGN OCL statement can be used to temporarily reassign the 5444 unit codes (the configuration record is not changed).
- All restrictions that apply to the assignment of 5444 unit codes and simulation areas during system generation also apply for $CNFIG. For these restrictions, see *Assigning Simulation Areas* in Part 3 of this manual.
- Only the 5444 unit codes and simulation areas included in the control statement are reassigned. All other previous assignments remain the same.

OCL CONSIDERATIONS

The following OCL statements are needed to load the configuration record program:



The code you supply depends on the location of the simulation area containing the configuration record program. Possible codes are F1, R1, F2, R2.

EXAMPLES

Figures 4-7 through 4-14.2 are examples of the control statements needed for various $CNFIG functions.

```
1    4    8    12   16   20   24   28   32   36
// ASNP2 R2-D2B,F2-D2A,R1-D1B,F1-D1A
// ASNP1 F1-D1A,R2-D2B,F2-D2A,R1-D1B
// ASNP3 R1-D1B,F1-D1A,R2-D2B,F2-D2A
// END
```

Explanation:

- The 5444 unit codes are reassigned for partition 2 (ASNP2).
- The 5444 unit codes are reassigned for partition 1 (ASNP1).
- The 5444 unit codes are reassigned for partition 3 (ASNP3).

Figure 4-7. Reassign All 5444 Unit Codes for Three Partitions

```
1    4    8    12   16   20   24   28   32   36
// PRINT
// ASNP2 F2-D2C
// PRINT
// END
```

Explanation:

- The current configuration record is printed (PRINT).
- The 5444 unit code F2 is reassigned to simulation area D2C (F2-D2C) for partition 2 (ASNP2).
- The revised configuration record is printed (PRINT).

Figure 4-8. Print the Current Configuration Record (PRINT), Reassign a Single 5444 Unit Code in Partition 2 (ASNP2), and Print the Revised Configuration Record (PRINT)

```
1    4    8    12   16   20   24   28   32   36
// TSTAMP I-YES,D-NO
// END
```

Explanation:

- All informational messages that appear on the system console are time-stamped when they are entered into the system history area (I-YES). Decision and reply messages and responses are not time-stamped (D-NO). The D-NO parameter can be specified or assumed.

Figure 4-9. Time-Stamp Only Informational Type Messages

```
1    4    8    12   16   20   24   28   32   36
// TSTAMP
// END
```

Explanation:

- No informational, decision, or reply messages and responses are time-stamped when entered into the system history area.

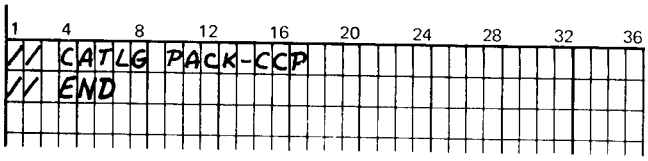
Figure 4-10. No Messages or Responses are Time-Stamped

```
1    4    8    12   16   20   24   28   32   36
// ASNP3 R2-D2D
// CATLG PACK-NONE
// END
```

Explanation:

- The 5444 unit code R2 is reassigned to simulation area D2D (R2-D2D) for partition 3 (ASNP3).
- The PACK-NONE parameter specifies that cataloging is not allowed to a simulation area being used as a program pack for another partition.

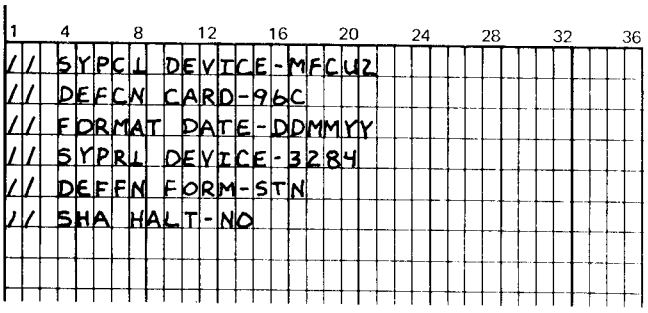
Figure 4-11. Reassign a 5444 Unit Code and Change the Support for Cataloging to a Program Pack



Explanation:

- The PACK-CCP parameter specifies that cataloging is allowed to a CCP program pack.

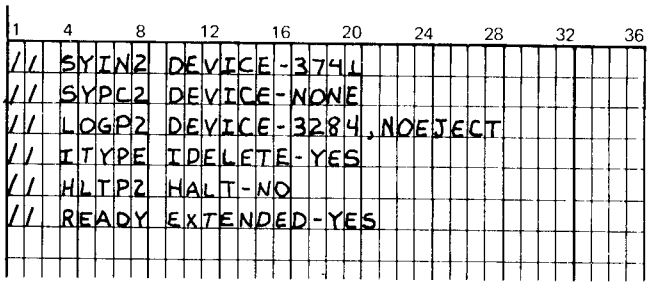
Figure 4-12. Change the Support for Cataloging to a Program Pack



Explanation:

- The system punch device for partition 1 (SYPC1) will be the 5424 secondary hopper (DEVICE-MFCU2).
- Card type 96C will be the spool default.
- The system date is to be entered in the format 24-08-77 (24 August 1977).
- The system print device for partition 1 (SYPR1) will be the 3284 Printer (DEVICE-3284).
- Form type STN will be the spool default.
- The system will no longer halt when the system history area is full.

Figure 4-13. Reassign System Devices and Output Media for Partition 1



Explanation:

- The system input device for partition 2 (SYIN2) will be the 3741 Data Station (DEVICE-3741).
- No system punch device (DEVICE-NONE) will be assigned to partition 2 (SYPC2).
- The log device for partition 2 (LOGP2) will be the 3284 Printer (DEVICE-3284). The printer page will not eject (NOEJECT) at end of step and end of job.
- With IDELETE-YES, HALT-NO, and EXTENDED-YES, the system operator is not required to respond to all system messages.

Figure 4-14. Reassign System and Log Devices for Partition 2 and Do Not Require Operator Intervention

```
1   4   8   12  16  20  24  28  32  36
// AUTST READ-YES,PUNCH-NO,PRINT-NO
// AUTWT PUNCH-YES,PRINT-YES
// QCOPY ERASE-NO
// QCOPY AUTHORIZE-YES,RQPTY-2
// SPCYL CYL-26
// SPDSK UNIT-D3
// SIZE  SYS-512,P1-196,FS-4,P3-0
// SHA   TRACKS-2
```

Explanation:

- The input spooling routine (READ-YES) automatically starts during an IPL (operator control commands are not required). The punch writer (PUNCH-NO) and the print writer (PRINT-NO) will not start until the operator control commands are entered.
- The punch writer (PUNCH-YES) and the print writer (PRINT-YES) automatically start (without operator control commands) when there is output on the queues.
- QCOPY information that is entered at a terminal under CCP is not erased from the terminal screen (ERASE-NO).
- Authorization (AUTHORIZE-YES) is required for those users who execute $QCOPY under control of CCP. The priority limit (RQPTY-2) of jobs placed on the active spool file reader queue by $QCOPY is 2 ($QCOPY must be executed under control of CCP).
- The number of cylinders assigned to the spool file is 25 (CYL-25).
- The spool file is located on the main data area coded D3 (UNIT-D3).
- The main storage size of the system is 512K (SYS-512); partition 1 size is 196K (P1-196); file share area size is 4K (FS-4); partition 3 size is zero (P3-0). Partition 2 is not specified and remains unchanged.
- The SHA warning message is issued and the system halts (if the system is in halt SHA mode) when 2 tracks remain in the system history area before unprinted entries are overlaid.

Figure 4-14.1. Change Spooling Status, QCOPY Options, System and Partition Sizes, and SHA Warning Point

```
1   4   8   12  16  20  24  28  32  36
// SPOPT M-YES
// CONSOL SHARE-NO
// MESSAG RETAIN-NO
// READY  MESSAGE-NO
// PRIORITY SEQUENCE-P1,P2-P3=SP
// SHA HALT-CCPAUTO,TRACKS-1
```

Explanation:

- The page/card count and the form/card type are included in the spool time messages that are entered in the system history area (M-YES).
- The console cannot be shared as a system input device among partitions (SHARE-NO).
- All I-type messages on the system console at end of job will be deleted (RETAIN-NO).
- Messages are not displayed on the console while ENTER READER DATA or ENTER DATA are prompted on the console (MESSAGE-NO).
- The priority of the partitions and spool is as follows: partition 1 (P1) has the highest priority, partition 2 (P2), partition 3 (P3), and SPOOL (SP) all have equal priority.
- The system will invoke $HACCP (HALT-CCPAUTO) when 1 track remains in the system history area (TRACKS-1) before unprinted entries are overlaid.

Figure 4-14.2. Change Console Sharing as System Input Device, System Messages, System Task Priority, and SHA Halt Status

This page intentionally left blank.

Copy/Dump Program—$COPY

PROGRAM DESCRIPTION

The copy/dump program has the following functions:

- Copy an entire main data area or simulation area
- Copy a data file
- Copy and print a data file
- Copy a data file, but print only a part of the file
- Print an entire data file
- Print only a part of a data file
- Print and copy a part of a data file
- Build an indexed file from a sequential file
- Build a direct file from a sequential file
- Recover a file

This program performs only one function during one execution.

The input file must be described in an OCL FILE statement except when the file recovery function is used. The name that you supply with the NAME keyword in the FILE statement must be COPYIN (NAME-COPYIN). Likewise, the output file (except printer output) must be described in an OCL FILE statement. The name that you supply with the NAME keyword must be COPYO (NAME-COPYO). The output file for the IBM 1403 Printer does not require an OCL FILE statement. However, if the output is printed on an IBM 3284 Printer, you must supply an OCL FILE statement. The name of the file must be COPYP (NAME-COPYP).

COPYPACK

The COPYPACK function copies the contents of a main data area to another main data area or copies a simulation area to another simulation area.

COPYFILE

The COPYFILE functions:

- Process a single input file and write a single output file in one execution of the program.
- Copy a file from a simulation or main data area to another simulation or main data area. Sequential files can be copied consecutively and an index created, thus effectively copying a sequential file to an indexed file. Also, a sequential file can be copied and a direct file created.
- Copy a file to or from magnetic tape, cards, or diskette.
- Print all or part of a file whether it is being copied to another file or not.
- Print and/or copy selected records from a file based on either the relative record number or a key value.
- Recover a file by using the physical address obtained from the VTOC listing.
- Copy a file and change the output file record length.

The selection of input and output device is as follows:

Input	Output										
	Disk File			Tape File (1)	Card File					Diskette File	Printer File (1403,3284)
	Seq.	Ind.	Dir.		MFCU1	MFCU2	MFCM1	MFCM2	1442		
Disk File											
Sequential	X	X	X	X	X	X	X	X	X	X	X
Indexed		X		X	X	X	X	X	X	X	X
Direct (2)			X	X	X	X	X	X	X	X	X
Tape File (1)	X	X	X	X (3)	X	X	X	X	X	X	X
Diskette File (4)	X	X	X	X	X	X	X	X	X		X
Card File											
MFCU1	X	X	X	X		X			X	X	X
MFCU2	X	X	X	X	X				X	X	X
MFCM1	X	X	X	X				X	X	X	X
MFCM2	X	X	X	X			X		X	X	X
1442	X	X	X	X	X	X	X	X		X	X
2501	X	X	X	X	X	X	X	X	X	X	X

(1) Tape can be 7- or 9-track; 200, 556, 800, or 1600 bpi.

(2) A direct file is considered a sequential file; therefore, the sequential disk file considerations apply.

(3) The output tape file must be on a different drive than the input tape file.

(4) Record length can be from 1 to 128 bytes but must be the same for all records in the file.

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

Functions ①	Control Statements ②
Copy an Entire Area	// COPYPACK FROM-code,TO-code[,PACKIN-packname] [,PACKO-packname] // END

Copy a Data File	// COPYFILE {OUTPTX- / OUTPUT-} {FILE, / DISK,} {DELETE- / OMIT-} 'position,character',REORG-{NO / YES} ③
	[,LENGTH-number]
	// END

Copy and Print a Data File	// COPYFILE {OUTPTX- / OUTPUT-} BOTH, {DELETE- / OMIT-} 'position,character',REORG-YES ③
	[,LENGTH-number]
	// END

Copy a Data File, But Print Only a Part of the File	// COPYFILE {OUTPTX- / OUTPUT-} BOTH, {DELETE- / OMIT-} 'position,character',REORG-YES ③
	[,LENGTH-number]
	// SELECT {KEY / PKY} {FROM-'key' / TO-'key'} [,TO-'key' / ,FROM-'key']
	// SELECT RECORD, {FROM-number / TO-number} [,TO-number / ,FROM-number]
	// END
	Only one SELECT statement for each COPYFILE statement

Print an Entire Data File	// COPYFILE {OUTPTX- / OUTPUT-} PRINT // END

Print Only a Part of a Data File	// COPYFILE {OUTPTX- / OUTPUT-} PRINT
	// SELECT {KEY / PKY} {FROM-'key' / TO-'key'} [,TO-'key' / ,FROM-'key']
	// SELECT RECORD, {FROM-number / TO-number} [,TO-number / ,FROM-number]
	// END
	Only one SELECT statement for each COPYFILE statement

Print and Copy a Part of a Data File	// COPYFILE {OUTPTX- / OUTPUT-} {FILE / DISK / BOTH / PRINT} [,LENGTH-number] ④ [,REORG-YES]
	// SELECT {KEY / PKY} {FROM-'key' / TO-'key'} [,TO-'key' / ,FROM-'key']
	// SELECT RECORD, {FROM-number / TO-number} [,TO-number / ,FROM-number] ,FILE-YES
	// END
	Only one SELECT statement for each COPYFILE statement

Build an Indexed File from a Sequential File	// COPYFILE {OUTPTX- / OUTPUT-} {FILE / DISK / BOTH} [,LENGTH-number]
	// KEY LENGTH-number, LOCATION-number
	// END

Functions	Control Statements

Build a Direct File from a Sequential File	// COPYFILE {OUTPTX- / OUTPUT-} {FILE / DISK / BOTH} [,LENGTH-number]
	// OUTDM DATAMGMT-DIRECT
	// END

File Recovery from Physical Address-Simulation Area	// COPYFILE {OUTPTX- / OUTPUT-} {FILE / DISK / BOTH} [,LENGTH-number]
	// ACCESS FROM-unit,CYLINDER-number,SECTOR-number,DISP-number,RECL-number
	// SELECT RECORD,FROM-number,TO-number,FILE-YES
	// END

File Recovery from Physical Address – Main Data Area	// COPYFILE {OUTPTX- / OUTPUT-} {FILE / DISK / BOTH} [,LENGTH-number]
	// ACCESS FROM-unit,CYLINDER-number,TRACK-number,SECTOR-number,DISP-number,
	RECL-number
	// SELECT RECORD,FROM-number,TO-number,FILE-YES
	// END

Change the Output File Record Length	// COPYFILE {OUTPTX- / OUTPUT-} {FILE / DISK / BOTH},LENGTH-number // END

① The program uses include the possible combinations of copying and printing files.

② For each use, the program requires the control statements in the order they are listed: COPYPACK, END; COPYFILE, END; COPYFILE,SELECT,END; COPYFILE,KEY,END; and COPYFILE,SELECT,KEY,END.

③ Applies only to indexed files. When OUTPUT-BOTH is specified, REORG-YES is required.

④ LENGTH parameter is not valid if OUTPTX-PRINT or OUTPUT-PRINT is specified.

PARAMETER SUMMARY

Parameter	Description
COPYPACK Statement	
FROM-code	Location of the area to be copied. Possible codes are R1, F1, R2, F2, and those for the main data areas.
TO-code	Location of the area to contain the copy. Possible codes are R1, F1, R2, F2, and those for the main data areas.
PACKIN-packname	Name of the area to be copied, optionally provided to check for pack ID. Verify that correct pack is mounted.
PACKO-packname	Name of output area, optionally provided to check for pack ID. Verify that correct pack is mounted.
COPYFILE Statement	
OUTPUT-FILE	Copy the file to the device (tape, cards, diskette, or disk) defined in the COPYO FILE statement. (1)
OUTPUT-DISK	Same as OUTPUT-FILE (allowed for Models 10 and 12 compatibility). (1)
OUTPUT-PRINT	Print the entire file or only part of the file. (1)
OUTPUT-BOTH (3)	Copy the file from one device to another or from one location to another location on the same area. (1) Also print the entire file or only part of it.
OUTPTX-{FILE, DISK, PRINT, BOTH}	Printed output is to be displayed in hexadecimal values.
DELETE-'position, character' or OMIT-'position, character'	These parameters are optional. All records with the specified character in the specified record position are deleted. DELETE causes deleted records to be printed; DELETE cannot be used with direct files. If OMIT is used, delete records are not printed. Position can be any position in the record (the first position is 1, second 2, and so on). The maximum position is 65535.
REORG-NO (2)	Indexed files only. Copy records as organized in the original file (the file from which the records are copied).
REORG-YES (2)(3)	Indexed files only. Reorganize the records so that the records in the data portion of the file are in the same order as their keys are listed in the index.
LENGTH-number	Used to specify the length of the output file record(s). Length value may be any number from 1 through 65535.

Parameter	Description
SELECT Statement	
{KEY / PKY},FROM-'key'	Indexed files only. Print or copy only the part of the file from the record key that is specified in the FROM parameter to the end of the file.
{KEY / PKY},TO-'key'	Indexed files only. Print or copy only that part of the file from the first record key to the key specified in the TO parameter.
{KEY / PKY},FROM-'key', TO-'key'	Indexed files only. Print or copy only the part of the file between the two record keys that are specified in the FROM and TO parameters (including the records indicated by the parameters). To print only one record, make the FROM and TO record keys the same.
RECORD,FROM-number	Print or copy only the part of the file from the relative record number specified in the FROM parameter to the end of the file.
RECORD,TO-number	Print or copy only that part of the file from the first record to the relative record number specified in the TO parameter.
RECORD,FROM-number, TO-number	Print or copy only the part of the file between the relative record numbers indicated by the parameters (including the records indicated by the parameter). To print only one record, the FROM and TO relative record numbers should be the same.
FILE-YES	Only selected records are copied to the files named in the COPYO FILE statement.
FILE-NO	Only selected records are printed. If copying, all records are copied. OUTPUT-PRINT or OUTPUT-BOTH must be specified if FILE-NO is specified. If OUTPUT-PRINT is specified, selected records are printed. If OUTPUT-BOTH is specified, selected records are printed and the entire file is copied to the file named in the COPYO FILE statement. When a KEY statement is used, the output will be an indexed file if the device on the COPYO FILE statement is a main data area.
KEY Statement	
LENGTH-number	Identifies the length of the key field. Key length may be 1-29.
LOCATION-number	The starting location in the input record from which the key field is to be extracted. Location may be from 1 to 65535.

Parameter	Description
OUTDM Statement	
DATAMGMT-DIRECT	Specifies that the output file is to be a direct file.
ACCESS Statement	
FROM-unit	Specifies the simulation area or main data area that contains the file to be recovered.
CYLINDER-number	Identifies the cylinder number (1-202) for the beginning of the file. For a simulation area, the number is the quotient obtained by dividing the file location (from the $LABEL VTOC printout) by 2. For a main data area, the number is given in the file location.
SECTOR-number	For a simulation area, the number can be 0-47. For a main data area, the number can be 1-48.
DISP-number	Specifies the displacement, in bytes, from the start of a sector to the beginning of a record in the same sector. Number can be 0-255.
TRACK-number	For a main data area. This number can be 0-19.
RECL-number	Record length of file to be recovered.

(1) In the OCL load sequence, you indicate which file is to be copied or printed. For files being copied, you must also indicate whether the file is being copied from one device to another or from one location to another on the same area, using the COPYIN and COPYO FILE statements.

(2) REORG-NO is assumed if you omit the REORG parameter. When OUTPUT-BOTH is used for indexed files, REORG-YES is required.

(3) If message UC3CCS occurs, indicating that there is not enough main storage available to execute the job, consider the following:

- If you have OUTPUT-BOTH, change to OUTPUT-DISK or OUTPUT-FILE.
- If you have REORG-YES, change to REORG-NO.
- Set a larger partition size if possible.

PARAMETER DESCRIPTIONS

FROM and TO Parameters (COPYPACK)

The FROM and TO parameters are used when the copy/dump program is copying the entire contents of one area to another. They indicate the locations of the two areas.

The FROM parameter (FROM-code) indicates the location of the area you are copying. The TO parameter (TO-code) indicates the location of the area that is to contain the copy. The FROM and TO codes must be for the same type of area (simulation or main data). You cannot copy a simulation area from or to a main data area.

Possible codes are R1, F1, R2, F2, and those for the main data areas (D1, D2, D3 or D31, D32, D33, D34, D4 or D41, D42, D43, D44).

Copying Entire Area

When copying an area, the copy/dump program transfers the contents of the area to another area. The contents of the two areas will be the same, except for the volume labels and alternate track information, which may be different.

The area you are copying can contain libraries or data files or both. The area that is to contain the copy must not contain libraries, temporary data files, or permanent data files.

Until the contents of the area are completely copied, portions of the new area are changed to prevent accidental usage of a partially filled area. Therefore, if the copying process is stopped before it is completed, the area is unusable. You can restart the copying process by reloading the copy/dump program, or you can restore the area by reinitializing.

After a successful copy, the copy program prints a message:

COPYPACK IS COMPLETE

Note: If you copy a simulation area containing an active checkpoint, that checkpoint will exist on both the FROM and TO simulation areas. When one of the two active checkpoints is utilized to restart the checkpointed program, care must be taken to ensure that the job is not restarted a second time. It is recommended that you perform an IPL and load Restart ($$RSTR) from the pack containing the second active checkpoint. If you then select the controlled cancel option when the HØnn message occurs (nn is the last requested checkpoint number), the checkpoint will be deactivated.

PACKIN and PACKO Parameters (COPYPACK)

These two optional parameters are used if you want the system to check the volume label of the areas. Either one or both may be specified. They are useful if you want to verify that the correct area(s) are being used.

COPYPACK Considerations

The following considerations apply when you use the COPYPACK statement:

- If you are copying files from a main data area on a 3344 to a main data area on a 3340, an IS message occurs if files are allocated on cylinders 167–186 of the 3344. Only cylinders 0–166 are copied to the 3340.
- If you are copying files from a main data area on a 3340 to a main data area on a 3344, only cylinders 0–166 are copied to the 3344.
- $COPY cannot copy an area that has active files unless they are files being accessed for input only.

OUTPUT Parameter (COPYFILE)

The OUTPUT parameter is used when the program is copying and printing card, tape, diskette, or disk data files. It indicates whether you want the program to copy, print, or copy and print a file. The OUTPTX parameter can be used to display printed output in hexadecimal values.

The parameter OUTPUT-DISK or OUTPUT-FILE is used to copy the file; OUTPUT-PRINT is used to print the file; and OUTPUT-BOTH is used to copy and print the file.

Copying Files

The copy/dump program can copy a file from one device to another. These devices can be disk, tape, card, or diskette. The copy/dump program can also copy a file from one location to another location on the same volume.

The OCL load sequence for the copy/dump program indicates (1) the name and location of the file being copied, and (2) the name and location of the copy being created. (See *OCL Considerations* in this section.)

In copying a file, the program can omit records. (See the description of the DELETE parameter for more information.)

In copying an indexed file, the program can reorganize records in the data portion such that they are in the same order as their keys are listed in the index. (See the description of the REORG parameter for more information.)

In copying an indexed file, the copy/dump program will:

- Copy the data and the entire index intact if you supply only the COPYFILE control statement and only the parameter OUTPUT-DISK or OUTPUT-FILE.
- Copy the data and create a new index if you supply more than one control statement (COPYFILE must be included and REORG-NO specified or assumed).
- Copy the data and create a new index if you supply only the COPYFILE control statement and the following parameters: OUTPUT-DISK or OUTPUT-FILE, REORG-NO (can be supplied or assumed), and one or more parameters that are applicable to the COPYFILE statement.

If you suspect that a problem exists with an index and the problem can be corrected by the COPYFILE control statement, you should code the control statement(s) so that the desired function is performed.

Printing Files

The program can print all or part of a data file. To print only part, the program needs a SELECT control statement. (See the description of the SELECT control statement parameters in this section.) If you do not use a SELECT statement, the entire file is printed.

If the output is to be printed on the 3284 Printer, a COPYP FILE statement must be entered. If the COPYP FILE statement is not entered, the output will be printed on the 1403 Printer.

If you use SELECT KEY (PKY), or REORG-YES, records from indexed files are printed in the order their keys appear in the index portion of the file; otherwise, they are printed as they appear in the file. For each record, the program prints the record key followed by the contents of the record.

Records from sequential and direct files are printed in the order they appear in the file. For each record, the program prints the relative record number followed by the contents of the record.

The program uses as many lines as it needs to print the contents of a record. Appendix A lists the hexadecimal representation for characters in the standard character set.

The following is an example of the way the program prints hexadecimal numbers using OUTPTX:

```
ABCDE GHIJ12345
CCCCCBCCCDFFFFF4444444
12345678911234500000000
```

The hexadecimal number B6 represents a character that has no print symbol.

After printing the last record, the printer triple spaces and prints the following message:

(number) RECORDS PRINTED

DELETE Parameter (COPYFILE)

In copying a data file, the copy/dump program can omit records of one type. The DELETE parameter identifies the type of record. Use of the DELETE parameter is optional. If you do not use it, no records are deleted. DELETE cannot be used with direct files.

The form of the parameter is DELETE-'position,character'. *Position* is the position of the character in the records. *Character* is the character, except for apostrophes, blanks, or commas, that identifies the record. For example, with the parameter DELETE-'100,R' all records with an R in position 100 are deleted. By specifying the hexadecimal code for the character, you can use any character (including apostrophes, blanks, commas, and packed data) to identify the records to be deleted. For example, with the parameter DELETE-'100,X40', all records with a blank (hexadecimal 40) in position 100 are deleted.

Deleted records are always printed. If you are both copying and printing a data file, deleted records are printed with the other records that are printed. The deleted records are preceded by the word DELETED.

The OMIT keyword can be used instead of DELETE. The deleted records are not printed if OMIT is used.

The records are deleted only from the copied file. The original file is not affected.

REORG (Reorganize) Parameter (COPYFILE)

In copying an indexed file, the program can reorganize the file so that the records in the data portion are in the same order as their keys in the file index. The REORG parameter indicates whether the file should be reorganized. If you want a file reorganized, use REORG-YES. Otherwise, use REORG-NO. REORG-NO is assumed if you omit the parameter.

If REORG-YES is specified, the reorganization applies to the copy of the file rather than the original file. The original file is not affected.

Reorganization (REORG-YES) is required when you are both copying and printing an indexed file (OUTPUT-BOTH).

LENGTH Parameter (COPYFILE)

This parameter is used to specify the length of the output records when a file is copied or the file type is changed. The resulting records will contain blanks or be truncated on the right. This parameter cannot be specified if the output is only printed.

KEY and PKY Parameters (SELECT)

The SELECT KEY and SELECT PKY parameters apply to selecting part of an indexed file. The SELECT PKY parameter applies to selecting part of an indexed file that contains packed keys. The parameters are FROM and TO.

The FROM parameter (FROM-'key') gives the key of the first record to be selected. The TO parameter (TO-'key') gives the key of the last record to be selected. The record keys between those two in the file index identify the remaining records to be selected. If you want to select only the one record, use the same record key in both the FROM and TO parameters.

For example, the parameters FROM-'000100' and TO-'000199' mean that records identified by keys 000100 through 000199 are to be selected.

If the file index does not contain the key you indicate in a FROM parameter, the program uses the next higher key in the index.

If the key in the FROM parameter is higher than the highest key in the indexed file, the highest key and record in the file will be printed (print-only function). For example, assume the following statement:

```
// SELECT KEY,FROM-'7000',TO-'7000'
```

If the highest key in the indexed file is 6955, then record 6955 is printed.

If the TO parameter is omitted, the program assumes that the last key in the index is the TO key.

If the FROM parameter is omitted, the program assumes that the first record key is the FROM key.

You can use fewer characters in the FROM or TO parameter than are contained in the actual keys; when keys are packed, however, you must use the same number of characters as contained in the actual keys. If you use fewer characters, the program ignores the remaining characters in the record key. The number of characters used in the FROM and TO parameters need not be the same.

For example, assume that the following are consecutive record keys in an index: A1000, A1119, A1275, A1900, A1995, A2075, and 99999. The parameters FROM-'A1' and TO-'A199' refer to record keys A1000 through A1995.

If none of the keys in the file index begin with the characters you indicate in a FROM parameter, the program uses the key beginning with the next higher characters in the FROM parameter.

For example, assume that four consecutive record keys in an index begin with these characters: A1, A2, A8, and B1. The parameters FROM-'A3' and TO-'A9' refer to keys beginning with the characters A8.

RECORD Parameters (SELECT)

The SELECT RECORD parameters can apply to any file but are normally used for sequential and direct files. These parameters use relative record numbers to identify the records to be selected.

Relative record numbers identify a record's location with respect to other records in the file. The relative record number of the first record is 1, the number of the second record is 2, and so on.

The SELECT RECORD parameters are FROM and TO. The FROM parameter (FROM-number) gives the relative record number of the first record to be selected. The TO parameter (TO-number) gives the number of the last record to be selected. Records between those two records in the file are also selected.

For example, the parameters FROM-1 and TO-30 mean that the first 30 records (1-30) in the file will be selected.

If the TO parameter is omitted, the program assumes that the number of the last record in the file is the TO number. If the FROM parameter is omitted, the program assumes that the first record in the file is the FROM number. If you want to select only one record, use the same number in the FROM and TO parameters.

Both parameters, FROM and TO, are required when the ACCESS statement is specified.

During a single execution of $COPY, the records of an indexed file cannot be both reorganized (REORG) and selected by relative record number (SELECT).

FILE Parameter (SELECT)

This parameter allows only selected records to be copied to a disk, tape, cards, diskette, or printer.

LENGTH and LOCATION Parameters (KEY)

The KEY statement is used when the program is to build an indexed file from a sequential file. The LENGTH parameter specifies the length (1–29) of the key field. The LOCATION parameter specifies the starting location (1–65535) of the key field in the input record. When the KEY statement is used, the file described in the COPYO FILE statement must be a disk file and OUTPUT-DISK, OUTPUT-FILE, or OUTPUT-BOTH must be specified in the COPYFILE control statement. A $INDEX45 or $INDEX40 work file can be included if the KEY statement is used.

DATAMGMT Parameter (OUTDM)

This parameter allows the creation of a direct file (disk) from sequential input (card, tape, disk, diskette).

FROM Parameter (ACCESS)

This parameter identifies the area (simulation or main data) that contains the file to be recovered. Possible unit codes are R1, F1, R2, F2, and those for the main data areas.

CYLINDER Parameter (ACCESS)

The cylinder number specified in this parameter indicates the starting location of the file. For a simulation area, the number can be 1–202 and is the quotient obtained by dividing the file location[1] by 2. For a main data area, the number can be 1–166 (186 for 3444) and is indicated by the file location[1].

SECTOR Parameter (ACCESS)

This parameter specifies the sector that contains the first record to be copied (recovered). For a simulation area, the number can be 0–47. For a main data area, the number can be 1–48.

[1] File location is obtained from the $LABEL printout of the VTOC.

TRACK Parameter (ACCESS)

This parameter is used for a main data area. The number can be 0–19 and is specified by the file location[1].

RECL Parameter (ACCESS)

This parameter identifies the record length of the data in the file to be recovered.

DISP Parameter (ACCESS)

This parameter specifies the displacement from byte 0 of a sector to the first byte of a record. Displacement is counted in bytes from the first byte (byte 0) of the sector. Because a sector contains 256 bytes, the displacement number must be 0–255.

COPYING MULTIVOLUME FILES

When you copy multivolume files, the first volume of the input file has to be online when the job is initiated. The output file must be a new file. If either condition is not satisfied, a message occurs.

Maintaining Proper Volume Sequence Numbers

To maintain proper volume sequence numbers when copying a multivolume file, you must either copy all the volumes of the file in one run or copy only one volume for each run of $COPY. For example, if you copy a three-volume file one volume at a time (volume 1 in the first run, volume 2 in the second run, and volume 3 in the third run), the volumes will retain their original sequence numbers in the output file. Or if you copy all the volumes (1, 2, and 3) in the same run, the volume sequence numbers in the new file will be the same as in the original file. However, if you copy only volumes 2 and 3 in one run, their volume sequence numbers will be changed to 1 and 2 in the output file.

$COPY ensures that all volumes of a multivolume file have the same date in the following manner. If only one volume of a multivolume file is copied for each run of $COPY, the new file will assume the same date as the input file. If all volumes or, as in the example above, volumes 2 and 3 of a three-volume file are copied in a single run, the new file will assume the current partition date.

Maintaining Correct Relative Record Numbers

To maintain correct relative record numbers when copying one volume of a multivolume direct file, you must keep the size of the output volume the same as the size of the input volume. (If you want to increase the size of a file, you must copy the entire file.) If you copy the first volume of a two-volume file and increase the number of records on that volume, you are also increasing relative record numbers of all the records on the next volume. Therefore, to maintain the correct relative record numbers, output and input volume extents (records or tracks) must be equal if you are copying only one volume of a multivolume direct file.

Direct File Attributes

If you copy an entire multivolume direct file in one run, the output file will be given sequential attributes in the volume table of contents (VTOC). However, this does not affect file processing. A file with either sequential or direct attributes can be accessed by a consecutive or random access method. If only one volume is copied, the direct attribute will be maintained.

Copying Multivolume Indexed Files

If you want to copy an indexed multivolume file, REORG-YES must be specified in the COPYFILE statement. Since an unordered load to a multivolume indexed file is not permitted, a REORG-NO causes a system message to be issued. If you would prefer not to reorganize the file it must be copied one volume at a time. When you are copying one volume at a time, the HIKEY on the output volume must be the same as the HIKEY on the input volume. If they are not equal, a system message occurs. Making the HIKEYs the same ensures that both the input and output volumes are the same length and no records will be lost. When you are copying one volume of a multivolume indexed file, either REORG-YES or REORG-NO may be specified.

[1] File location is obtained from the $LABEL printout of the VTOC.

TAPE FILE CONSIDERATIONS

When copying or printing tape data files, you must describe the tape file being copied or printed and describe the file being created. The various tape record formats and labels are supported.

$COPY supports single volume tape files, multivolume tape files, and multifile tape volumes. For a detailed description of the FILE statement parameters, see the appropriate FILE statement in Part 1 of this manual.

The tape file can be ASCII or EBCDIC. Default for record format (RECFM) is fixed length. On a nonlabeled tape, record length (RECL) and block length (BLKL) must be specified.

$COPY will not copy any data records from a null file to an unlabeled tape. Therefore, when files are added to a multifile tape and a null file is detected, the sequence number for the next file(s) added to the tape is 1 less than the value specified. See *Null Files On Tape* for additional information.

You must be careful when copying a tape file with variable length records to disk or tape. The resulting file will contain fixed length records with a record length equal to the longest record length of the file copied from. Records copied with short record lengths will have invalid information in the unused portion of the output record.

DISKETTE FILE CONSIDERATIONS

When copying or printing diskette data files, you must describe the diskette file being copied or printed and the file being created. (See *FILE Statement [Device Independent Files]*.)

The RECL parameter identifies the record length for the diskette file and is any number from 1 to 128. If this parameter (RECL) is not used, the default is 96.

When the 3741 is used as an input or output device, the number specified in the RECL parameter must be equal to the record length in the data set label on the diskette and may be any number from 1 through 128.

When the 3741 is used for output and the input is from disk, card, or tape, the number specified in the RECL parameter can be any number from 1 to 128 regardless of the record length of the disk, card, or tape file being copied. However, if the record length from disk, card, or tape is less than the record length specified, the remainder of the record is filled with blanks (X'40'). If the record length from disk, card, or tape is greater than the record length specified in the RECL parameter, the record is truncated on the right.

Note: The following card input considerations also apply for the 3741.

CARD INPUT CONSIDERATIONS

For card input files, end-of-file is determined by the presence of a card with /* in columns 1 and 2, and with the remaining columns blank. This allows a card input file to contain /* cards, assuming that at least one punch is in columns 3–80 or 3–96. A /& or /. is handled the same as a /* card, unless the input device is the system reader.

The following chart shows the results obtained from having a card with a /*, /&, or /. in columns 1 and 2.

Card Columns 1 and 2	Input Device		Spooled	Not Spooled	Card Has Comments	Card Does Not Have Comments	Results
	Partition Reader	Not Partition Reader					
/*	X		X		X		Card is processed as a data card
/*	X		X			X	End of file occurs
/*	X			X	X		Card is processed as a data card
/*	X			X		X	End of file occurs
/*		X	X		X		Card is processed as a data card
/*		X	X			X	End of file occurs
/*		X		X	X		Card is processed as a data card
/*		X		X		X	End of file occurs
/&	X		X		X		Card is accepted as data but end of file occurs (additional data cards are not processed)
/&	X		X			X	End of file occurs
/&	X			X	X		Card is accepted as data but end of file occurs (additional data cards are not processed)
/&	X			X		X	End of file occurs
/&		X	X		X		Card is accepted as data card
/&		X	X			X	End of file occurs
/&		X		X	X		Card is accepted as data card
/&		X		X		X	End of file occurs
/.	X		X		X		Card is accepted as data but end of file occurs (additional data cards are not processed)
/.	X		X			X	End of file occurs
/.	X			X	X		Card is accepted as data but end of file occurs (additional data cards are not processed)
/.	X			X		X	End of file occurs
/.		X	X		X		Card is accepted as data card
/.		X	X			X	End of file occurs
/.		X		X	X		Card is accepted as data card
/.		X		X		X	End of file occurs

Note: Reading from the spool reader queue that is not the partition reader is not recommended.

CARD OUTPUT CONSIDERATIONS

If the input record size (in bytes) is greater than the size of the card (80 or 96 columns), the input record will be truncated. Only the first 80 or 96 positions of the record will be punched. If the input record size is less than the size of the card, the unused part of the card will contain blanks. If the input file contains 60-byte records, the card will be blank in columns 61–80 or 61–96. With punched output, the entire card contains user data; control information is not punched.

FILE RECOVERY CONSIDERATIONS

The ACCESS and SELECT control statements are used to recover indexed, direct, and sequential files. Information regarding the record count, record length, file location, and start of data (indexed files only) for each file to be recovered is obtained from the current VTOC printout. For information regarding the VTOC printout, refer to *File and Volume Label Display Program–$LABEL.*

The method for determining the first recoverable record location is generally trial and error. You should always attempt to recover the first record of the file. If unsuccessful, proceed to the next record. Repeat this procedure until the first recoverable record is found.

When records are added to a sequential disk file, the VTOC is updated at end of job with a new end-of-file pointer. If, for some reason, the program adding the records is abnormally terminated, the VTOC is not updated and the added records are lost. However, you can use the copy/dump program to retrieve the added records as follows:

- Use the ACCESS control statement to specify the beginning of the file, and
- Use the SELECT control statement to specify the number of records in the original file plus the number of added records.

This procedure causes the original records plus the retrieved records to be copied to a new file and a new VTOC entry created.

Note: Do not include a FILE statement with NAME-COPYIN specified when the ACCESS statement is used.

For information about how to copy a file index or create a new file index, refer to *Copying Files* under *Copy/Dump Program.*

OCL CONSIDERATIONS

The following OCL statements are needed to load the copy/dump program, if you are using the program to copy an entire area:

```
// LOAD $COPY,code
// RUN
```

The code you supply depends on the location of the simulation area containing the copy/dump program. Possible codes are R1, F1, R2, F2.

The following OCL statements are needed to do COPYFILE functions:

```
// LOAD $COPY,code
// FILE NAME-COPYIN,parameters      (required except when the ACCESS statement is used)
// FILE NAME-COPYO,parameters       (optional statement)
// FILE NAME-COPYP,parameters       (optional statement)
// RUN
```

For information on the FILE statement parameters, see *OCL Statements* in Part 1 of this manual.

The UNIT parameter is required on each entered FILE statement. The allowable UNIT codes are:

Unit Code	FILE Statement		
	COPYIN	COPYO	COPYP
3741	X	X	
MFCU1	X	X	
MFCU2	X	X	
MFCM1	X	X	
MFCM2	X	X	
1442	X	X	
2501	X		
1403			X
3284			X
R1, F1, R2, F2	X	X	
D1, D2, D3 or D31	X	X	
D32, D33, D34	X	X	
D4 or D41	X	X	
D42, D43, D44	X	X	
T1, T2, T3, T4	X	X	

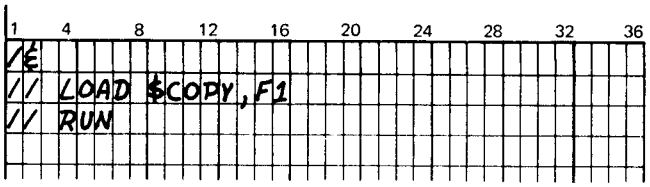
Note: During execution of $COPY, the 3741 or 1442 cannot be assigned both input and output.

EXAMPLES

Figures 4-15 through 4-20 are examples of the OCL statements and control statements needed to copy an entire area, copy a file from one area to another area, and print part of a file.

Figures 4-21 through 4-37 are examples of the OCL statements and control statements needed to:

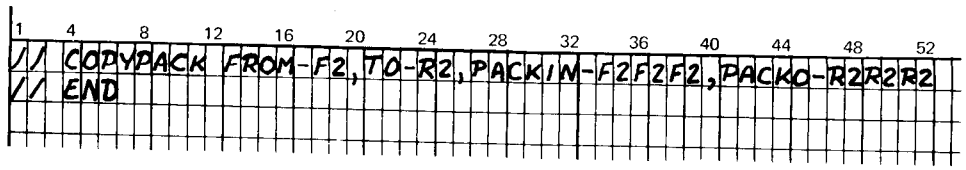
- Copy a file from disk to tape
- Copy a file from tape to disk, printing part of the file
- Copy a file from tape to tape, selecting records to be copied
- Copy a card file to tape
- Copy a disk file to cards
- Copy a disk file to diskette
- Copy a tape file to diskette, printing part of the file
- Copy and print a portion of a file from diskette to disk
- Copy a card file to a diskette, printing the entire file
- Copy and print a portion of a file from diskette to cards
- Copy a card file to another card file
- Print a disk file on the 3284
- Copy a card file to a disk file and change the output record length
- Build a direct disk file from sequential tape input
- Recover a disk file from a simulation area
- Recover a disk file from a main data area
- Copy a sequential file from a simulation area to a main data area and create an indexed output file



Explanation:

The copy/dump program is loaded from F1.

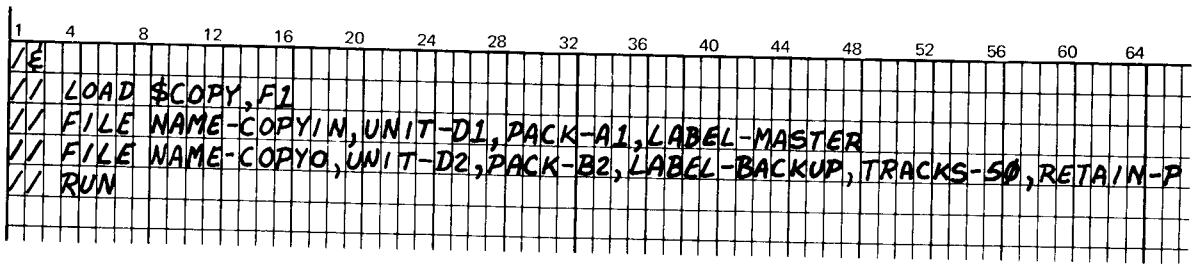
Figure 4-15. OCL Load Sequence for Copying an Entire Area



Explanation:

The COPYPACK statement copies the contents of simulation area F2 (FROM-F2) with volume identification F2F2F2 (PACKIN-F2F2F2) onto simulation area R2 (TO-R2) with volume identification R2R2R2 (PACKO-R2R2R2).

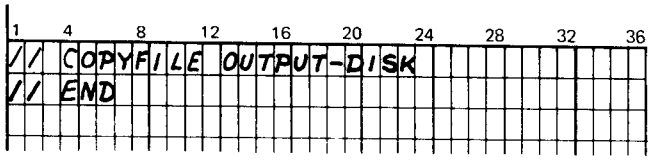
Figure 4-16. Control Statements for Copying an Area



Explanation:

- Copy/dump program is loaded from F1.
- Input file (OCL sequence):
  - The name that identifies file is MASTER (LABEL-MASTER).
  - Main data area D1 contains the file (UNIT-D1). Its name is A1 (PACK-A1).
- Output file (OCL sequence):
  - The name to be written to identify the file is BACKUP (LABEL-BACKUP).
  - The area that is to contain the file is the main data area D2 (UNIT-D2). The pack name is B2 (PACK-B2).
  - The file is to be permanent (RETAIN-P).
  - The size of the file is 50 tracks (TRACKS-50).

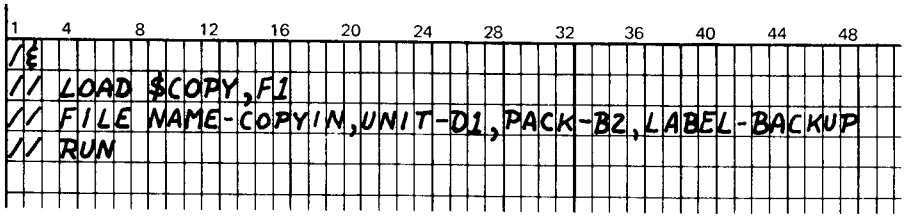
Figure 4-17. OCL Load Sequence for Copying a File from One Area to Another



Explanation:

The COPYFILE statement tells the program to create the output file using all the data from the input file. The output file is a copy of the input file.

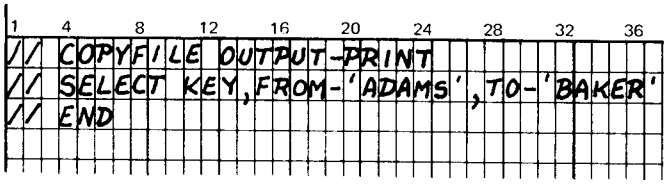
Figure 4-18. Control Statements for Copying a File from One Area to Another



Explanation:

- Copy/dump program is loaded from F1.
- Input file (OCL sequence):
  - The name that identifies the file is BACKUP (LABEL-BACKUP).
  - The area that contains the file is the main data area on drive 1 (UNIT-D1). Its name is B2 (PACK-B2).

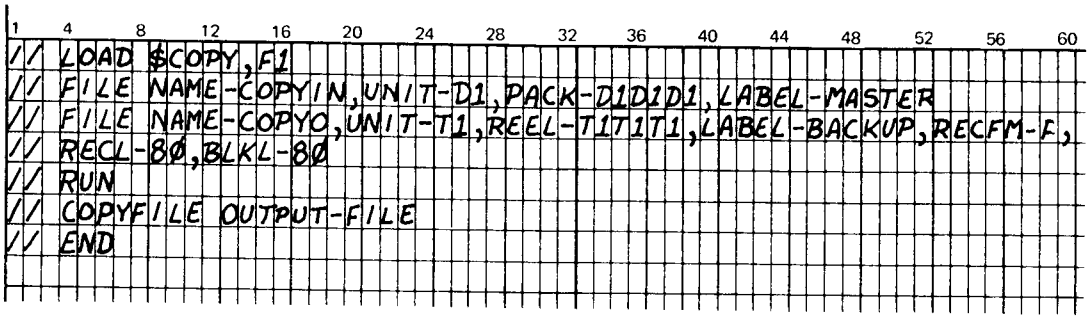
Figure 4-19. OCL Load Sequence for Printing Part of a File



Explanation:

- The file is being printed (COPYFILE statement).
- The file is an indexed file. The part being printed is identified by the record keys from ADAMS to BAKER in the index (SELECT statement).

Figure 4-20. Control Statements for Printing Part of a File



Explanation:

- Copy/dump program is loaded from F1.

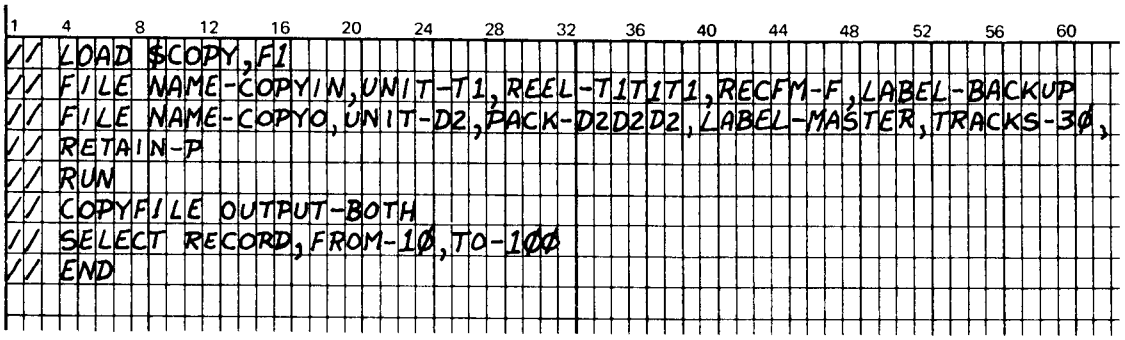
- Input file (OCL sequence):
  - The name that identifies the file is MASTER (LABEL-MASTER).
  - The area that contains the file is the main data area on drive 1 (UNIT-D1). Its name is D1D1D1 (PACK-D1D1D1).

- Output file (OCL sequence):
  - The name to be written on tape to identify the file is BACKUP (LABEL-BACKUP).
  - The tape unit that is to contain the file is tape unit 1 (UNIT-T1). Its name is T1T1T1 (REEL-T1T1T1).
  - The record format used is fixed length, unblocked records (RECFM-F). The record length is 80 (RECL-80).

- Control statement explanation:

  The entire file named MASTER is copied to tape unit 1 (OUTPUT-FILE).

Figure 4-21. OCL and Control Statements to Copy a Disk File to a Tape File



Explanation:

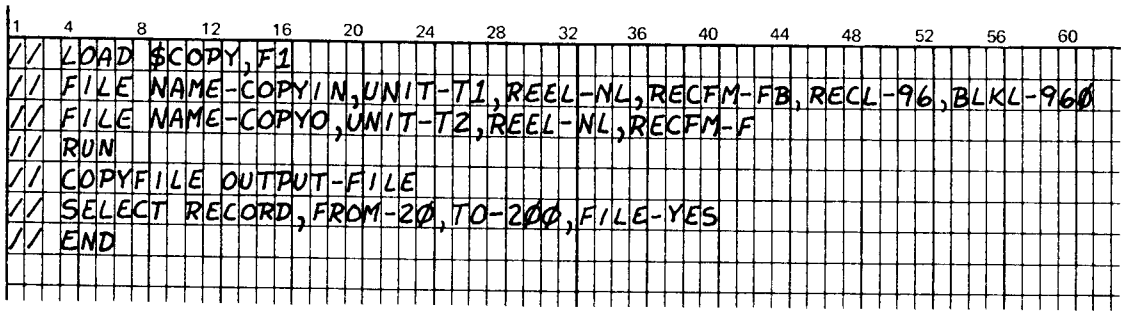
- Copy/dump program is loaded from F1.

- Input file (OCL sequence):
  - The name that identifies the file on tape is BACKUP (LABEL-BACKUP).
  - The tape that contains the file is tape unit 1 (UNIT-T1). Its name is T1T1T1 (REEL-T1T1T1).
  - The record format of the file is fixed length, unblocked records (RECFM-F).

- Output file (OCL sequence):
  - The name to be written to identify the file is MASTER (LABEL-MASTER).
  - The area that is to contain the file is the main data area on drive 2 (UNIT-D2). Its name is D2D2D2 (PACK-D2D2D2).
  - The file is to be permanent (RETAIN-P).
  - The size of the file is 30 tracks (TRACKS-30).

- Control statement explanation:
  - The entire file is copied from tape to disk (OUTPUT-BOTH).
  - The records 10 through 100 are printed (RECORD, FROM-10, TO-100).

Figure 4-22. OCL and Control Statements to Copy a Tape File to a Disk File and Print a Part of the File



Explanation:

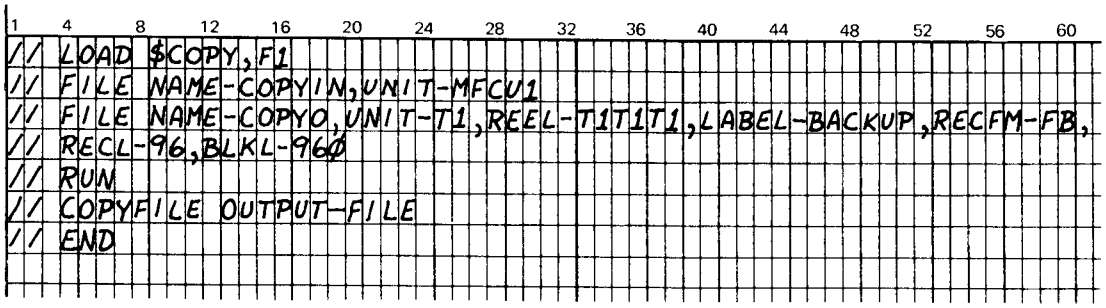
- Copy/dump program is loaded from F1.

- Input file (OCL sequence):
  - The tape that contains the file is tape unit 1 (UNIT-T1).
  - The tape being copied is an unlabeled tape (REEL-NL); therefore, record format (RECFM-FB), record length (RECL-96), and block length (BLKL-960) are specified.

- Output file (OCL sequence):
  - The tape unit that is to contain the file is tape unit 2 (UNIT-T2).
  - No label is used on the output tape (REEL-NL).
  - The record format is fixed length, unblocked (RECFM-F).

- Control statement explanation:
  - Records 20 to 200 are copied (FILE-YES).
  - No records are printed (OUTPUT-FILE).

Figure 4-23. OCL and Control Statements to Copy a Tape File to a Tape File and Select Records to be Copied



Explanation:

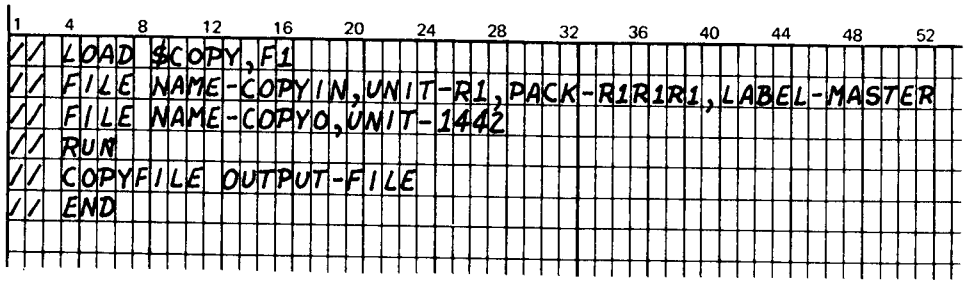
- The copy/dump program is loaded from F1.
- Input file (OCL sequence)

  The primary hopper of the 5424 MFCU contains the input file (UNIT-MFCU1).

- Output file (OCL sequence):
  - The name to be written on tape to identify the file is BACKUP (LABEL-BACKUP).
  - The tape unit that will contain the file is tape unit 1 (UNIT-T1). Its name is T1T1T1 (REEL-T1T1T1).
  - The record format used is fixed length, blocked records (RECFM-FB). The record length will be 96 (RECL-96); the block length will be 960 (BLKL-960).
- Control statement explanation:

  The entire card file will be copied to tape unit 1 (OUTPUT-FILE).

Figure 4-24. OCL and Control Statements to Copy a Card File to a Tape File



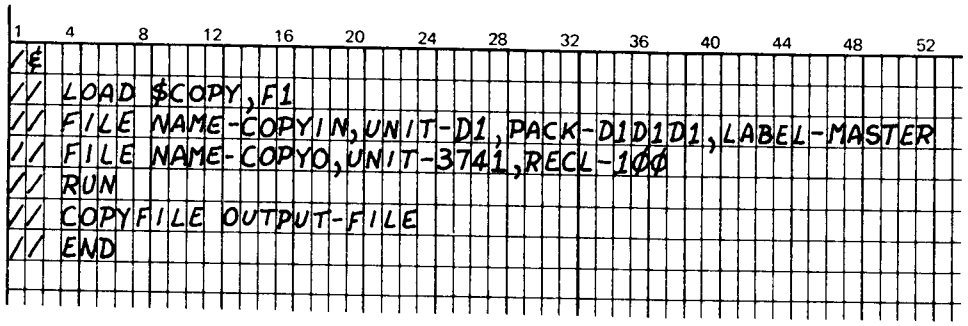
Explanation:

- The copy/dump program is loaded from F1.
- Input file (OCL sequence):
  - The name that identifies the file is MASTER (LABEL-MASTER).
  - The area that contains the input file is R1 (UNIT-R1). The area name is R1R1R1 (PACK-R1R1R1).
- Output file (OCL sequence):

  The 1442 will contain the output file (UNIT-1442).
- Control statement explanation:

  The entire disk file named MASTER will be punched into cards by the 1442 (OUTPUT-FILE).

Figure 4-25. OCL and Control Statements to Copy a Disk File to a Card File



Explanation:

- The copy/dump program is loaded from F1.

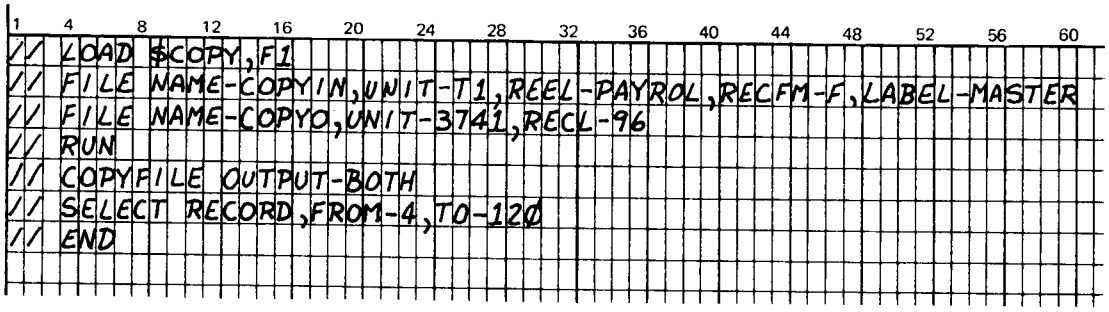
- Input file (OCL sequence):
  - The name that identifies the file is MASTER (LABEL-MASTER).
  - The area that contains the file is D1 (UNIT-D1). Its name is D1D1D1 (PACK-D1D1D1).

- Output file (OCL sequence):
  - The output file is contained on diskette (UNIT-3741).
  - The record length specified in the 3741 data set label is 100 (RECL-100).

- Control statement explanation:

  The entire disk file named MASTER will be copied to the 3741 diskette (OUTPUT-FILE).

Figure 4-26. OCL and Control Statements to Copy a Disk File to a 3741 Diskette



Explanation:

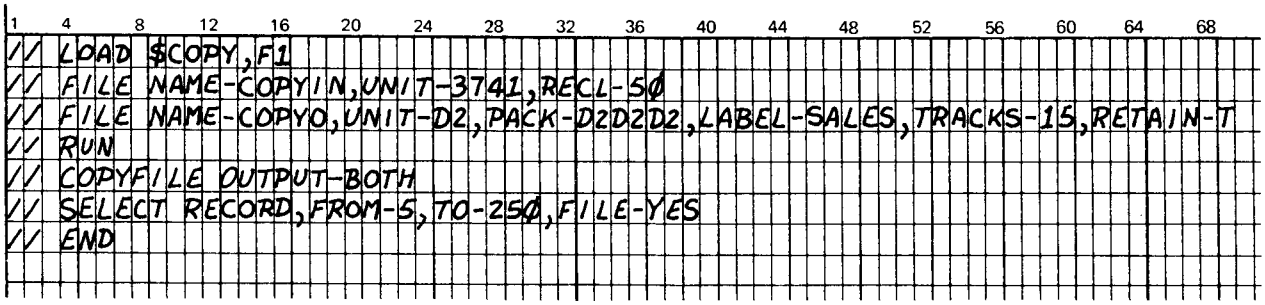
- The copy/dump program is loaded from F1.

- Input file (OCL sequence):
  - The name that identifies the file on tape is MASTER (LABEL-MASTER).
  - The tape that contains the file is tape unit 1 (UNIT-T1). Its name is PAYROL (REEL-PAYROL).
  - The record format of the file is fixed length, unblocked records (RECFM-F).

- Output file (OCL sequence):
  - The output file is contained on diskette (UNIT-3741).
  - The record length specified in the 3741 data set label is 96 (RECL-96).

- Control statements explanation:
  - The entire file is copied from tape to the 3741 (OUTPUT-BOTH).
  - Records 4 through 120 are printed (RECORD,FROM-4,TO-120).

Figure 4-27. OCL and Control Statements to Copy a Tape File to a Diskette File and Print a Part of the File



Explanation:

- The copy/dump program is loaded from F1.

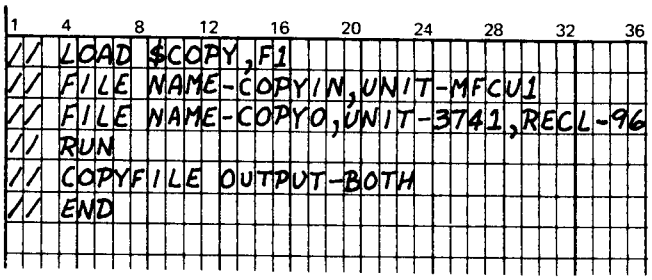
- Input file (OCL sequence):
  - The input file is contained on diskette (UNIT-3741).
  - The record length specified in the 3741 data set label is 50 (RECL-50).

- Output file (OCL sequence):
  - The name of the output file is SALES (LABEL-SALES).
  - The area that is to contain the file is main data area on drive 2 (UNIT-D2). Its name is D2D2D2 (PACK-D2D2D2).
  - The file is to be temporary (RETAIN-T).
  - The size of the file is 15 tracks (TRACKS-15).

- Control statements explanation:

  Records 5 to 250 are copied (FILE-YES) and printed (OUTPUT-BOTH).

Figure 4-28. OCL and Control Statements to Copy a Diskette File to a Disk File and Print Only the Copied Records



Explanation:

- Copy/dump program is loaded from F1.
- Input file (OCL sequence):

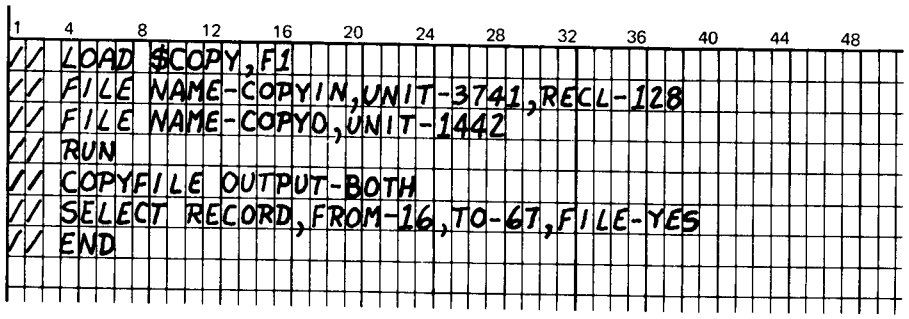
  The primary hopper of MFCU (5424) contains the input file (UNIT-MFCU1).

- Output file (OCL statement):
  - The output file is contained on diskette (UNIT-3741).
  - The record length specified in the 3741 data set label is 96 (RECL-96).

- Control statement explanation:

  The entire card file from the MFCU1 is copied to the 3741 and printed (OUTPUT-BOTH).

Figure 4-29. Control Statement to Copy a Card File to a Diskette and Print the Entire File



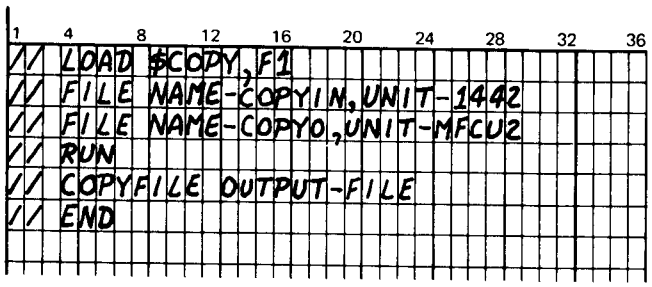
Explanation:

- Copy/dump program is loaded from F1.
- Input file (OCL sequence):
  - The input file is contained on diskette (UNIT-3741).
  - The record length specified in the 3741 data set label is 128 (RECL-128).
- Output file (OCL sequence):

  The 1442 contains the output file (UNIT-1442).
- Control statement explanation:

  Records 16 through 67 are copied (FILE-YES) to the 1442 and printed (OUTPUT-BOTH).

Figure 4-30. Control Statements to Copy and Print a Portion of a File on a Diskette to a Card Device



Explanation:

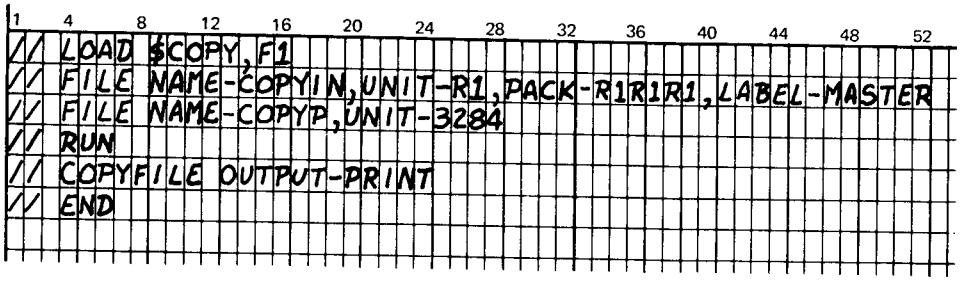
- Copy/dump program is loaded from F1.
- Input file (OCL sequence):

  The 1442 contains the input file (UNIT-1442).
- Output file (OCL sequence):

  The secondary hopper of the MFCU contains the output file (UNIT-MFCU2).
- Control statement explanation:

  The entire file is copied to the MFCU2.

Figure 4-31. Control Statement to Copy a Card File to Another Card File



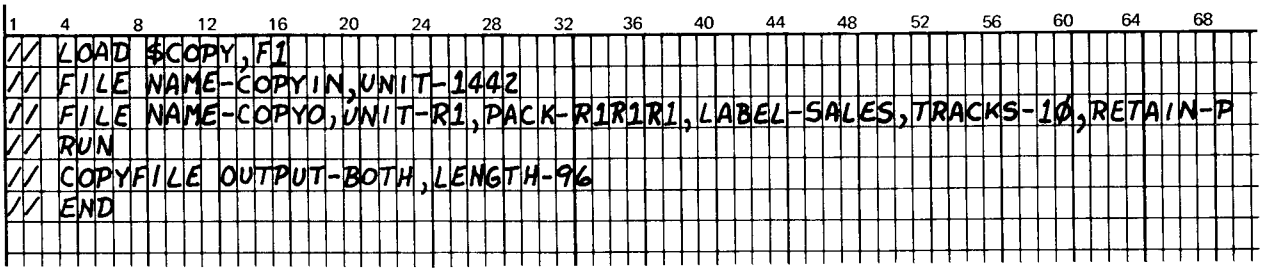
Explanation:

- The copy/dump program is loaded from F1.
- Input file (OCL sequence):
  - The name that identifies the file is MASTER (LABEL-MASTER).
  - The area that contains the file is the simulation area R1 (UNIT-R1). Its name is R1R1R1 (PACK-R1R1R1).
- Output file (OCL sequence):

  The unit that will contain the file is the 3284 (UNIT-3284).
- Control statement explanation:

  The entire disk file named MASTER will be printed on unit 3284 (OUTPUT-PRINT).

Figure 4-32. OCL Control Statement to Print a Disk File on the 3284



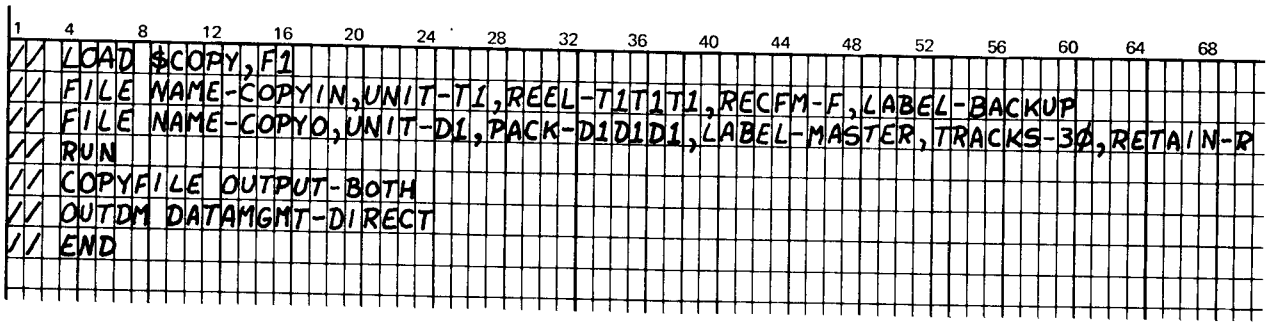
Explanation:

- The copy/dump program is loaded from F1.
- Input file (OCL sequence):

  The 1442 contains the input file (UNIT-1442).
- Output file (OCL sequence):
  - The name to be written to identify the file is SALES (LABEL-SALES).
  - The area that is to contain the file is the simulation area R1 (UNIT-R1). Its name is R1R1R1 (PACK-R1R1R1).
  - The file is to be permanent (RETAIN-P).
  - The size of the file is 10 tracks (TRACKS-10).
- Control statement explanation:

  The entire file will be copied to disk and printed (OUTPUT-BOTH) and the output record length will be 96 (LENGTH-96). Without the length parameter, output record length would be 80 (same as for 1442).

Figure 4-33. OCL and Control Statements to Copy a Card File to a Disk File and Specify an Output Record Length



Explanation:

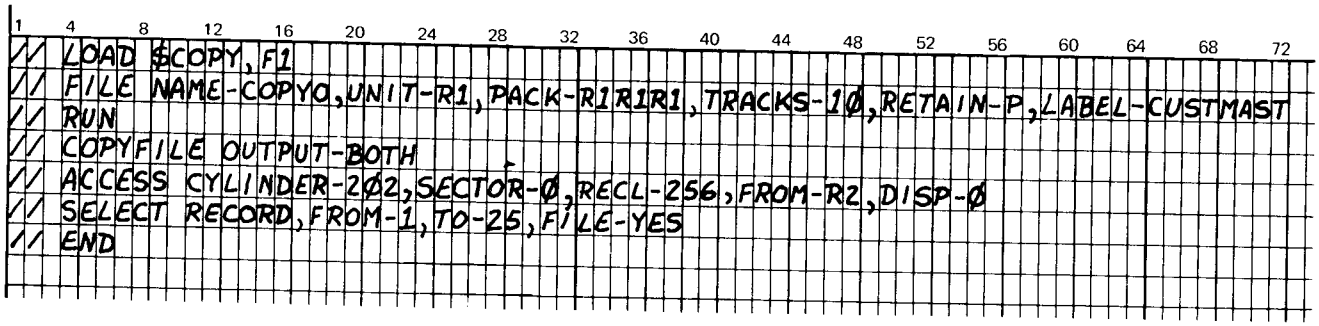
- Copy/dump is loaded from F1.

- Input file (OCL sequence):
  - The name that identifies the file on tape is BACKUP (LABEL-BACKUP).
  - The tape that contains the file is on tape unit 1 (UNIT-T1); its name is T1T1T1 (REEL-T1T1T1).
  - The record format of the file is fixed length, unblocked records (RECFM-F).

- Output file (OCL sequence):
  - The name to be written to identify the file is MASTER (LABEL-MASTER).
  - The area that is to contain the file is main data area D1 (UNIT-D1); its name is D1D1D1 (PACK-D1D1D1).
  - The file is to be permanent (RETAIN-P).
  - The size of the file is 30 tracks (TRACKS-30).

- Control statement explanation:
  - The entire file will be copied from tape to disk and printed (OUTPUT-BOTH).
  - The output file will be direct (DATAMGMT-DIRECT).

Figure 4-34. OCL and Control Statements to Build a Direct Disk File from Sequential Tape Input



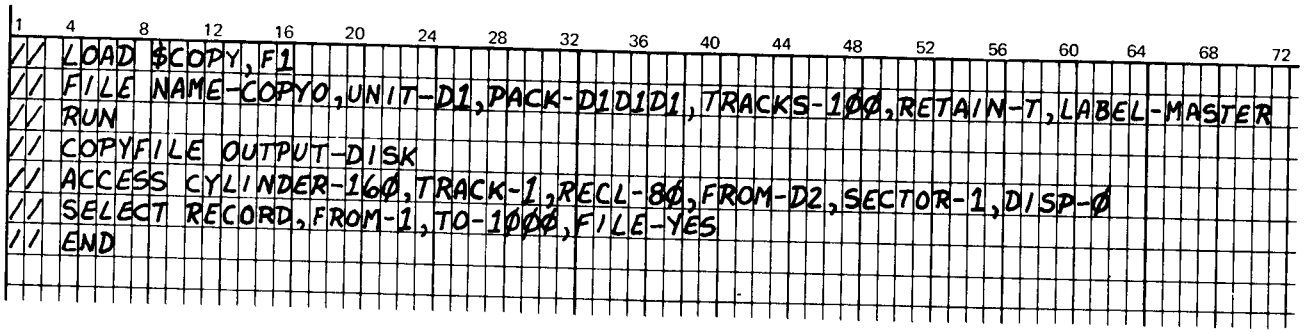
Explanation:

- The copy/dump program is loaded from F1.

- Output file (OCL sequence):
  - The name to be written to identify the file is CUSTMAST (LABEL-CUSTMAST).
  - The area that is to contain the file is simulation area R1 (UNIT-R1); its name is R1R1R1 (PACK-R1R1R1).
  - The file is to be permanent (RETAIN-P).
  - The size of the file is 10 tracks (TRACKS-10).

- Control statement explanation:
  - The file will be copied from cylinder 202 (CYLINDER-202), sector 0 (SECTOR-0); the record length of the input file is 256 bytes per record (RECL-256); the file is on simulation area R2 (FROM-R2), and the record starts with the first byte of the sector (DISP-0).
  - Records 1 through 25 will be copied and printed (FROM-1,TO-25, FILE-YES).

Figure 4-35. OCL and Control Statements to Recover a Disk File from a Simulation Area



Explanation:

- The copy/dump program is loaded from F1.

- Output file (OCL sequence):
  - Name to be written to identify the file is MASTER (LABEL-MASTER).
  - The main data area that is to contain the file is on drive 1 (UNIT-D1). The area name is D1D1D1 (PACK-D1D1D1).
  - The file is to be temporary (RETAIN-T).
  - The size of the file is 100 tracks (TRACKS-100).

- Control statement explanation:
  - The file will be copied from cylinder 160 (CYLINDER-160), track 1 (TRACK-1), the record length of the file is 80 bytes (RECL-80), the file is recovered from main data area D2 (FROM-D2), and the record starts with the first byte of sector 1 (SECTOR-1, DISP-0).
  - Records 1 through 1000 will be copied (FROM-1, TO-1000, FILE-YES).

Figure 4-36. OCL and Control Statements to Recover a Disk File from a Main Data Area

```
1   4    8    12   16   20   24   28   32   36   40   44   48   52   56   60
// LOAD $COPY,F1
// FILE NAME-COPYIN,UNIT-R1,PACK-R1R1R1,LABEL-CONSVF
// FILE NAME-COPYO,UNIT-D1,PACK-D1D1D1,TRACKS-100,LABEL-INDSVF
// FILE NAME-$INDEX45,UNIT-D2,PACK-D2D2D2,TRACKS-100,RETAIN-S
// RUN
// COPYFILE OUTPUT-DISK
// KEY LENGTH-23,LOCATION-128
// END
```

Explanation:

- Copy/dump program is loaded from F1.

- Input file (OCL sequence):
  - The name that identifies the file is CONSVF (LABEL-CONSVF).
  - The area that contains the file is simulation area R1 (UNIT-R1). Its name is R1R1R1 (PACK-R1R1R1).

- Output file (OCL sequence):
  - The name of the output file is INDSVF (LABEL-INDSVF).
  - The area that is to contain the file is the main data area on drive 1 (UNIT-D1). Its name is D1D1D1 (PACK-D1D1D1).
  - The size of the file is 100 tracks (TRACKS-100).

- Workfile

  $INDEX45 is a work file that may decrease execution time if a large indexed file is being created. For performance reasons, the work file should not be on the same disk drive as the output file.

- The COPYFILE statement tells the program to create an index for the output file using all the data from the input file.

- The KEY statement tells the program to create an index for the output file consisting of 23-byte keys (LENGTH-23) that are located 128 bytes into the record (LOCATION-128).

Figure 4-37. OCL and Control Statements to Copy a Sequential File From a Simulation Area to a Main Data Area and Create an Indexed Output File

Dump/Restore Program—$DCOPY

PROGRAM DESCRIPTION

The dump/restore program has the following functions:

- Dump the entire contents of a main data area or simulation area onto magnetic tape, or dump the entire contents of a simulation area onto diskette.
- Restore the main data area or simulation area to its original contents by transferring information back from the tape or diskette.

Important areas, such as those containing libraries and permanent data files, are normally copied. The tape or diskette contains a copy of all tracks and serves as a backup copy in case something happens to the information on the disk.

An active spool file in the input area is dumped to tape. However, during the restore operation, the spool file is deleted.

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

Functions	Control Statements ①
Copy an entire area to tape or diskette; or restore an area from tape or diskette.	// COPYPACK ② {TO-code / FROM-code} [,PACK-name] [,SYSTEM-{code / NO / YES}] [,BACKUP-{TAPE / 3741}] // END ③

① Control statements are required in the order they are listed.

② There can be only one COPYPACK statement in a program.

③ END statement must appear only once in a program since it is a delimiter indicating end of job.

PARAMETER SUMMARY

COPYPACK Statement

Parameter	Meaning
FROM-code	Location of area to be copied. Possible codes are F1, R1, F2, R2, and those for the main data areas.
TO-code	Location of area to receive the copy. Possible codes are F1, R1, F2, R2, and those for the main data areas. See Figure 4-38 for relationship of FROM and TO locations.
PACK-name	Name of the main data area or simulation area being used.
SYSTEM-NO	The SYSTEM-NO parameter does not allow cylinder 0 IPL areas to be dumped or restored.
SYSTEM-YES	SYSTEM-YES specifies that the IPL areas on cylinder 0 are to be dumped or restored along with the specified simulation area.
SYSTEM-code	Dump the IPL areas from cylinder 0 of the main data area along with the simulation area specified by the FROM parameter. Possible codes are those for the main data areas.
BACKUP-TAPE	The BACKUP-TAPE parameter specifies that magnetic tape (3410-3411) is to be used for dump/restore.
BACKUP-3741	BACKUP-3741 specifies that diskettes are to be used to dump or restore the specified simulation area.

PARAMETER DESCRIPTIONS

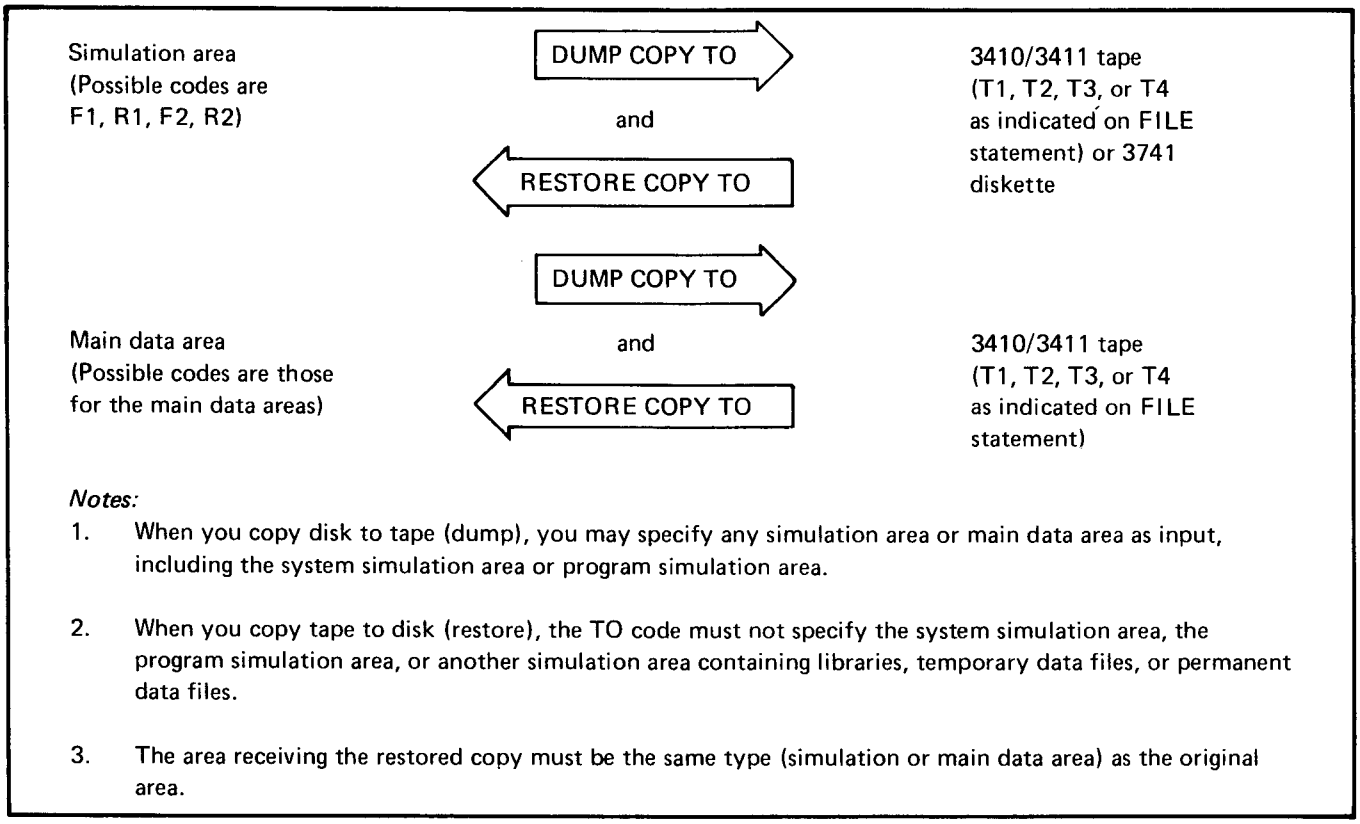
FROM and TO Parameters (COPYPACK)

The COPYPACK statement is used to copy information from disk to tape, tape to disk, disk to diskette, or diskette to disk.

The FROM parameter (FROM-code) indicates the location of the area being copied. The TO parameter (TO-code) indicates the location of area to receive the copy.

Possible codes for the FROM and TO parameters are R1, F1, R2, F2, and those for the main data areas.

See Figure 4-38 for the relationship of FROM and TO locations.



Notes:

1. When you copy disk to tape (dump), you may specify any simulation area or main data area as input, including the system simulation area or program simulation area.

2. When you copy tape to disk (restore), the TO code must not specify the system simulation area, the program simulation area, or another simulation area containing libraries, temporary data files, or permanent data files.

3. The area receiving the restored copy must be the same type (simulation or main data area) as the original area.

Figure 4-38. Relationship of Disk to Tape Drives When Using $DCOPY

PACK Parameter (COPYPACK)

The pack name specified is checked against the actual name of the main data area or simulation area. A halt occurs if they are not the same. If the parameter is not used, no checking occurs.

SYSTEM Parameter (COPYPACK)

The SYSTEM parameter is an optional parameter used to specify whether cylinder 0 IPL information is to be dumped or restored with the specified simulation area. SYSTEM-YES allows cylinder 0 to be dumped or restored to either tape or diskette. SYSTEM-code allows the cylinder 0 IPL areas of the unit specified by the code along with the simulation area specified in the FROM parameter to be dumped to either tape or diskette. SYSTEM-NO does not allow cylinder 0 to be dumped or restored. The default is SYSTEM-NO.

If SYSTEM-YES is specified for the dump, SYSTEM-YES must also be specified for the restore.

BACKUP Parameter (COPYPACK)

The BACKUP parameter specifies which device (tape or diskette) is to be used for backup. Tape may be used to back up a main data area, a simulation area, and cylinder 0. Diskettes can be used only to back up a simulation area and cylinder 0. Also, the 3741 data set must be set for 128-byte records.

DUMP/RESTORE CONSIDERATIONS

When you dump a main data area on a 3340 to tape and then restore the tape to a main data area on a 3344, only cylinders 0–166 are copied to the 3344.

When you dump a main data area on a 3344 to tape and then restore the tape to a main data area on a 3340, only cylinders 0–166 are copied to the 3340.

$DCOPY can copy areas with active files if the active files are input only. Areas cannot be copied if they contain active files that are being modified.

A 'WF' message occurs if an attempt is made to restore a main data area on a 3340 from a tape that contains a dump from a main data area on a 3344 that has files allocated on cylinders 167–186. Only cylinders 0–166 can be copied to the 3340.

During the restore function, portions of the new area (simulation or main data) are changed to prevent accidental usage of a partially filled area until the contents of the tape or diskettes are completely copied to the new area. Therefore, if the copying process is stopped before it is completed, the area is unusable. You can restart the copying process by reloading the dump/restore program, or you can restore the area by reinitializing.

Before you dump a main data area from a 3344, you should:

1. Determine if an active file(s) exists on cylinders 167 through 186 (if necessary, execute $LABEL). If so, you should execute the File Compress Program ($FCOMP) to move this file(s) within cylinders 1 through 166 (this occurs only if sufficient space exists on these cylinders).

2. Execute the Dump/Restore Program ($DCOPY). This program then copies (dumps) only cylinders 0 through 166.

OCL CONSIDERATIONS

The $DCOPY system service program requires the following OCL statements:

```
// LOAD $DCOPY,code

// FILE parameters

// RUN
```

The code identifying the location of the $DCOPY program can be R1, F1, R2, or F2.

FILE Statement Considerations

- The name of the file must always be BACKUP.
- When a 7-track tape is used for the dump/restore program, CONVERT-ON must be specified.
- The record format is always fixed length.
- The END position of the tape after processing always defaults to UNLOAD.
- During a restore operation, the density parameter must be the same number as specified for the dump.
- The record length, if specified, is ignored because $DCOPY makes the record length equal to the block length.
- The FILE statement is not required when the program is copying from disk to diskette.
- $DCOPY supports single volume tape files, multivolume tape files, and multifile tape volumes. For a detailed description of the FILE statement parameters, see the appropriate FILE statement in Part 1 of this manual.

Statement Entries

Statement Entry	Consideration
// LOAD	None
$DCOPY	Name of dump/restore program.
code	Location of simulation area containing dump/restore program. Possible codes are R1, F1, R2, or F2.
// FILE	None
NAME-filename	Filename entry must be BACKUP.
BLKL-block length	Block length and record length must be equal and one of the following values: *Note:* The tape record created is 2 bytes longer than specified because a 2-byte logical record number is appended to the tape record. Defaults are underlined.

Disk	Length in Bytes	Number of Tracks
Simulation area	<u>3072</u>	1/2 track
	6144	1 track
	12288	2 tracks
Main data area	<u>3072</u>	1/4 track
	6144	1/2 track
	12288	1 track
	24576	2 tracks

// RUN	None

For a detailed description of the OCL statements, see Part 1 of this manual.

Note: The remaining FILE statement parameters are described in the tape file OCL statement in Part 1.

Messages for DUMP/RESTORE

Message	Meaning
COPYPACK IS COMPLETE	This message indicates that the area has been dumped to tape or the tape has been restored to disk.
N TRACKS NOT RESTORED AT { CC/SS / CCC/HH/RR }	This message indicates the tracks that have not been restored on the simulation area or main data area. N = the number of tracks not restored. CC/SS is the address for a simulation area. CCC/HH/RR is the address for a main data area.
NN TAPE ERRORS OCCURRED PACK IS NOT COMPLETELY RESTORED.	This message indicates that tape errors have occurred or the restored area has missing data. NN = the number of tape errors. See previous messages for location of tracks not restored.

EXAMPLES

The parameters of the FILE statement vary depending upon whether the copy is to or from the tape.

Figures 4-39 through 4-44 show the OCL and control statements to dump from disk to tape, restore from tape to disk, and dump from disk to diskette.

FILE Statement: From Disk to Tape

Only required parameters are included in this example. See *OCL Considerations* for a listing of possible parameters.

```
// LOAD $DCOPY,F1
// FILE NAME-BACKUP,UNIT-T2
// RUN
```

Explanation:

- The dump/restore program is loaded from F1.
- The file name is always BACKUP (NAME-BACKUP).
- The copy goes to tape unit 2 (UNIT-T2).
- Tape unit 2 is a 9-track drive.

Figure 4-39. LOAD and FILE Statements to Dump from Disk to Tape

Control Statements

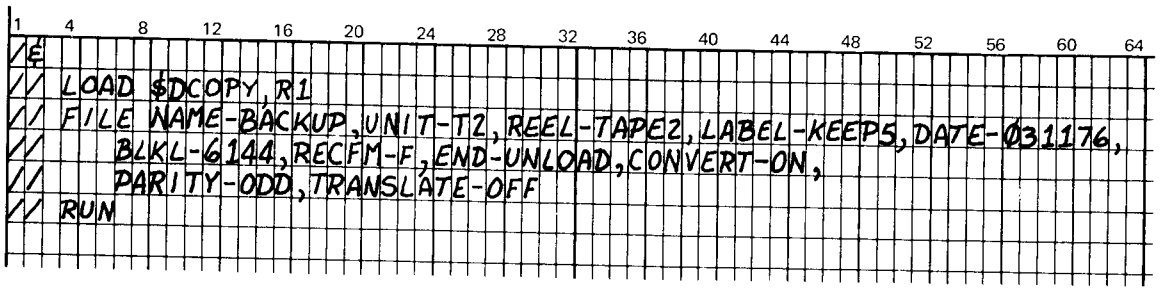
```
// COPYPACK FROM-F1,PACK-FIXED1
// END
```

Explanation:

- The COPYPACK statement tells the program to copy an entire area to tape.
- The copy is from F1 (FROM-F1).
- FIXED1 is the name of the simulation area being used (PACK-FIXED1). The program verifies that the area has the volume label FIXED1.

Figure 4-40. Control Statement to Dump from Disk to Tape

FILE Statement: From Tape to Disk



Explanation:

- The dump/restore program is loaded from R1.
- The file name is always BACKUP.
- Tape unit 2 contains the area copy.
- Tape unit 2 is a 7-track drive.
- TAPE2 is the label of the tape volume.
- KEEP5 is used in the header label.
- The date is March 11, 1976.
- Block length is 6144.
- CONVERT-ON indicates data conversion.
- END, PARITY, and TRANSLATE parameters given are the same as the default values.

Figure 4-41. LOAD and FILE Statements to Restore from Tape to Disk

```
1   4    8   12   16   20   24   28   32   36   40   44   48   52
// COPYPACK TO-F1,PACK-FIXED1,SYSTEM-YES,BACKUP-TAPE
// END
```

Explanation:

- The COPYPACK statement tells the program to copy an entire tape to F1 (TO-F1).
- The program restores cylinder 0 IPL records along with F1 (SYSTEM-YES).
- FIXED1 is the name of the simulation area being used (PACK-FIXED1). The program verifies that the area has the volume label FIXED1.
- Backup medium is magnetic tape.

Figure 4-42. Control Statement to Restore from Tape to Disk

Control Statement: From Disk to Diskette

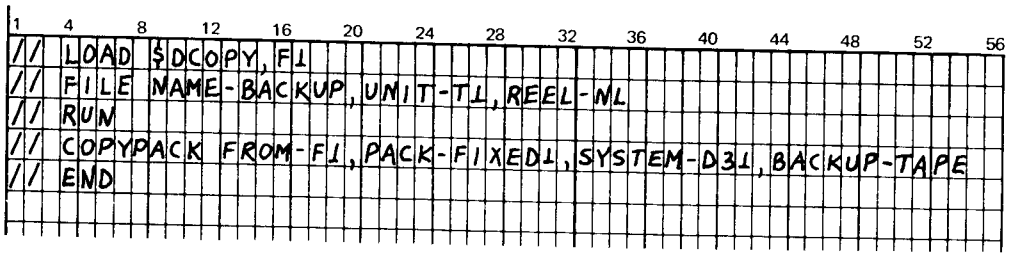
```
1   4    8   12   16   20   24   28   32   36
// LOAD $DCOPY,F1
// RUN
// COPYPACK FROM-F2,BACKUP-3741
// END
```

Explanation:

- The dump/restore program is loaded from F1.
- The COPYPACK statement tells the program to copy the simulation area F2 (FROM-F2) to the 3741 (BACKUP-3741).
- Approximately 11 diskettes are needed to contain the copy from simulation area F2.
- The record length on the 3741 diskette must be 128.

Figure 4-43. LOAD and Control Statements to Dump from Disk to Diskette

Control Statement: From Disk to Tape



Explanation:

- The dump/restore program is loaded from F1.
- The COPYPACK statement tells the program to copy the simulation area F1 (FROM-F1).
- The program copies cylinder 0 IPL records from D31 along with F1 (SYSTEM-D31).
- FIXED1 is the name of the simulation area being used (PACK-FIXED1). The program verifies that the area has the volume label FIXED1.
- Backup medium is magnetic tape (BACKUP-TAPE).

Figure 4-44. Load and Control Statements to Dump from Disk to Tape

Programming Considerations

When dumping from one of the simulation areas to diskette, put the 3741 online in mode 3. (Modes 1, 2, and 5 result in extent error conditions at the end of each diskette.) See note.

When restoring from diskette to one of the simulation areas, put the 3741 online in mode 3 or mode 5. If the 3741 is put online in mode 1, $DCOPY goes to end of job at the end of the first diskette. If the 3741 is put online in mode 2, the operator must put the 3741 online after each diskette is read (see note).

The COPYPACK IS COMPLETE message indicates the successful completion of $DCOPY. If this message is not logged after restoring to disk, the simulation area copied to will not be usable.

Note: Refer to *IBM System/3 3741 Reference Manual,* GC21-5113, for further explanations of the 3741 modes of operation.

File Delete Program—$DELET

PROGRAM DESCRIPTION

The file delete program has the following functions:

- Remove all file information from the VTOC and the data from an area.
- Remove file information by name from the VTOC and the data from an area.
- Remove file information in the volume table of contents (VTOC) only. This frees the space it occupies for use by new files but does not remove the data from an area.
- Free space that has been allocated but, due to abnormal circumstances, does not actually contain data.

Deleting Files

The program may be used for temporary and permanent files. To delete permanent files, you must use the file delete program. You can delete temporary files by using the file delete program or by changing the file designation from temporary to scratch (using the OCL keyword RETAIN) when you use the file.

When a REMOVE statement is used, file information is erased from the VTOC. The REMOVE statement can also be used to erase data from an area.

To be compatible with System/3 Models 8, 10, and 12, Model 15 recognizes the SCRATCH statement. However, all functions are performed as if it were a REMOVE statement.

Freeing Space on an Area

When a program creates a new file on an area, space for the file is allocated when the program is loaded. At the end of the program, the VTOC on that area is updated to reflect the creation of the new file.

Under very unusual circumstances, such as power failure or re-IPL while the program is running, the system may be prevented from updating the VTOC even though space has already been allocated for the new file. If this happens, the file delete program may be used to free this space.

Merely canceling a program or responding to a halt with an immediate cancel option does NOT cause this problem to occur because the system in this case frees up the unused space.

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

Functions	Control Statements (1)
Remove all files from an area	// REMOVE PACK-name, UNIT-code, LABEL-VTOC [, DATA-{NO / YES}]
Remove only the file(s) named from an area	// REMOVE PACK-name, UNIT-code, LABEL-{filename / 'filenames'} DATE-date (2) [,DATA-{NO / YES}]

Free all allocated space not containing files, libraries, or system areas	(3)
	// FORMAT UNIT-code, PACK-name
	// END

(1) For each use, the program requires the control statements in the order they are listed: REMOVE, END. Defaults are underlined.

(2) Use this form of the REMOVE statement when two or more files have the same name and you want to delete one of them.

(3) Use this control statement when you suspect that a system failure or an inadvertent re-IPL may have left space allocated but not actually being used.

PARAMETER SUMMARY

Parameter	Description	
PACK-name	Name of the area.	
UNIT-code	Location of the area. Possible codes are R1, F1, R2, F2, and those for the main data areas.	
LABEL-VTOC	Remove all file information from the VTOC.	
LABEL-filename	Remove only the VTOC information for the specified file.	Use names that identify files in VTOC. (1)
LABEL-'filename,filename,...	Remove only the VTOC information for the specified files.	
DATE-date	Date on the file being deleted. Date must be a six-digit number. Example: DATE-060776 means June 7, 1976	
DATA-YES NO	Delete data as well as VTOC entry. Do not delete the data; delete only the VTOC entry.	

(1) These are the names you gave the files when you placed them on an area.

PARAMETER DESCRIPTIONS

PACK Parameter

The PACK parameter (PACK-name) tells the program the name of the area that contains the files being deleted. The name you supply in this parameter is the one written by the disk initialization program.

For a simulation area, it is the name assigned by the simulation area program $COPY.

The file delete program compares the name in the PACK parameter with the volume label to ensure they match. In this way, the program ensures that it is using the right area.

UNIT Parameter

The UNIT parameter (UNIT-code) tells the program the location of the area containing the files being deleted. Possible codes are R1, F1, R2, F2, and those for the main data areas.

LABEL Parameter

The LABEL parameter identifies the file(s) you want to delete from the area. Its form depends on the file(s) you are deleting:

Form	Files Deleted
LABEL-VTOC	All of them.
LABEL-filename	Only the file that is named. The name can apply to more than one file. If it does, all of those *files are deleted* unless you use a DATE parameter to identify a particular one.
LABEL-'filename, filename,...'	Only the files that are named. A name can apply to more than one file. If it does, all of those files are deleted. Files that you want to delete using the DATA parameter must be in separate control statements. You can list as many file names as the statement can hold; the statement length, however, is restricted to 80 or 96 characters. Additional REMOVE statements may be used for additional file names.

DATE Parameter

The DATE parameter can be used only with LABEL-filename. The DATE parameter (DATE-date) applies to two or more files that have the same name. It tells the program the date of the file you want to delete.

Every file has a date, which is given to the file at the time it is created. When two or more files have the same name, the dates are used to distinguish one file from another.

If the area has more than one file with the name you list in the LABEL parameter, they will be deleted unless you use the DATE keyword and parameter to indicate a particular file. If the DATE keyword is used, only one filename can be given in the LABEL parameter for the control statement.

The date is a six-digit number in the form mmddyy or ddmmyy (d-day, m-month, y-year) Be sure to specify the date in the form specified during system generation.

In the DATE parameter, be sure to specify day, month, and year in the same order as they were specified when you created the file.

DATA Parameter

The DATA parameter allows you to erase the file data as well as the VTOC entry.

If YES is coded in this parameter, the data in the file specified is removed and any reference to it in the VTOC is also removed. In addition, the following is issued for each file removed.

```
'DATA REMOVED FOR FILE XXXXXX
DATA 000000'
```

DATA-YES should be used only if file security is required. The time needed to remove the data is much greater than the time needed to remove only the VTOC entry.

If NO is coded in this parameter, only the VTOC entry for the file specified is removed. If this parameter is not used, DATA-NO is assumed.

OCL CONSIDERATIONS

The following OCL statements are needed to load the file delete program:

```
// LOAD $DELET,code
// RUN
```

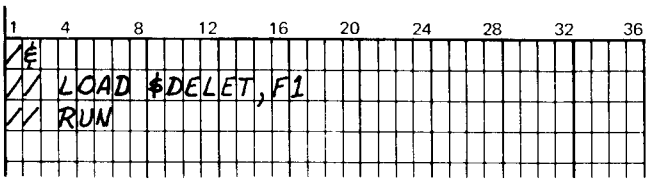
The code you supply depends on the location of the simulation area containing the program. Possible codes are R1, F1, R2, F2.

EXAMPLES

Deleting One of Several Files Having the Same Name

Figures 4-45, 4-46, 4-47, and 4-48 show the OCL statements and control statements needed to delete one of several files having the same name.

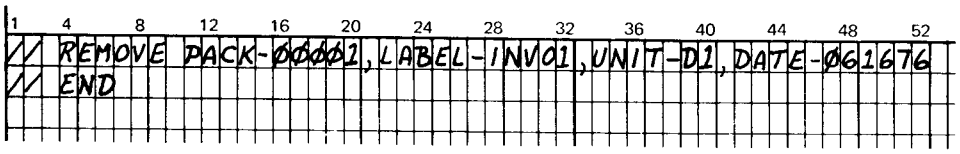
Assume that three files on a removable disk have the same name: INVO1. The dates of these files are 6/16/76, 6/18/76, and 1/15/76. You want to delete the version dated 6/16/76.



Explanation:

- The file delete program is loaded from F1.

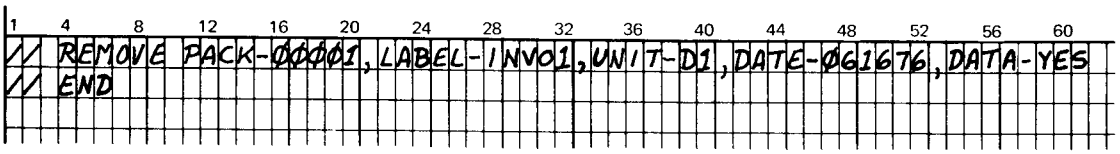
Figure 4-45. OCL Load Sequence for File Delete



Explanation:

- The area that contains the file being deleted is named 00001 (PACK-00001 in REMOVE statement).
- Because two other files have the name INVO1, the date (061676) is needed to complete the identification of the file you want to delete (LABEL-INVO1 and DATE-061676).
- The main data area containing the file to be deleted is D1 (UNIT-D1).

Figure 4-46. Control Statement to Delete One Version of a File



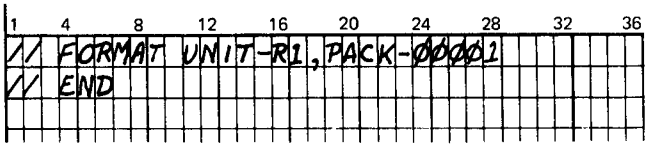
Explanation:

- The area that contains the file being deleted is named 00001 (PACK-00001 in REMOVE statement).
- Because two other files have the name INVO1, the date (061676) is needed to complete the identification of the file you want to delete (LABEL-INVO1 and DATE-061676).
- The main data area containing the file to be deleted is D1 (UNIT-D1).
- The YES specification in the DATA parameter deletes all data from the area containing information on the specified file.

Figure 4-47. Control Statement to Delete One Version of a File and Its Data

Freeing Allocated But Unused Space on an Area

Figure 4-48 shows the FORMAT control statement. The following control statement frees any areas on the simulation area R1 that have been allocated but are not being used. This condition may exist following the abnormal termination (such as a power failure or re-IPL) of a program that was creating a file.



Explanation:

- Free any allocated but unused space on the simulation area R1 (UNIT-R1) named 00001 (PACK-00001).

Figure 4-48. Control Statement to Free Allocated But Unused Space on a Simulation Area

File Compress Program—$FCOMP

PROGRAM DESCRIPTION

The file compress program has the following functions:

- Copy files from one main data area to another main data area without altering the files in either area.
- Move files within the same main data area.
- Back up (disk to tape) all files or selected files contained on a main data area(s).
- Restore (tape to disk) all files or selected files to a main data area(s).

The file compress program performs only one function (copy, move, backup, or restore) during one execution.

MOVE AND COPY FUNCTIONS

The code you supply in the FROM and TO parameters determines which function is performed. If the same main data area code is specified in each of the parameters, the program performs the move (compress) function. If you supply different main data area codes in the FROM and TO parameters, the program performs the copy function.

Move Function

During the move function, the program removes gaps from between files by moving each file so that it occupies disk space adjacent to the previous file. If COMPRESS-LO is specified or assumed, after $FCOMP is executed the files in the specified main data area occupy a contiguous space starting at cylinder 1; if COMPRESS-HI is specified, after execution the files in the specified main data area occupy a contiguous space that ends at the highest numbered available cylinder (see note).

Copy Function

The copy function copies (adds) the files on the main data area specified by the FROM code to the existing files on the main data area specified by the TO code.

If COMPRESS-HI is specified, space for the added files is allocated beginning from the highest numbered available cylinder; otherwise, space for the added files is allocated from the beginning of the main data area (cylinder 1).

Note: The file compress program attempts to move all files into a contiguous space that either starts at cylinder 1 or ends at cylinder 166 (3340) or cylinder 186 (3344). However, the file compress program will not move or copy the $SPOOL or the $CCPDUMP file; in addition, if the system's measurement facility is active, the file compress program will not move a $MONITOR file. Therefore, if an exception file ($SPOOL, $CCPDUMP, or $MONITOR) is encountered during the move function, gaps may exist between the last record (first record if COMPRESS-HI is specified) of the compressed files and the beginning (end if COMPRESS-HI is specified) of each of the exception files.

BACKUP AND RESTORE FUNCTIONS

The FROM or TO parameter, which you specify on a control statement, determines the function that is performed by the program. If you specify the FROM parameter, the program performs the backup function; if you specify the TO parameter, the program performs the restore function. However, the program will perform only one function during one execution of the program. Therefore, you cannot intermix control statements that have FROM and TO parameters. You can, however, specify a maximum of 10 control statements during one execution of the program.

Backup Function

During the backup function, the tape is created as a multifile volume. The first block of information on the tape contains $FCOMP information about the file. The remaining blocks are data blocks. The file(s) is blocked to tape with the blocking factor specified on an OCL FILE statement.

Each file that is copied is assigned a number. This number enables the program to locate the file if you choose to restore only selected files.

After a file has been copied, the FILE xxxxxxxx HAS BEEN COPIED TO POSITION zzzz ON TAPE message is issued.

After a successful backup function, the TAPE COPY IS COMPLETE message is issued.

The COMPRESS keyword is ignored if specified on the backup function.

Restore Function

During the restore function, the file(s) is placed in the first available space beginning from cylinder 1 (COMPRESS-LO), cylinder 166 (COMPRESS-HI and 3340), or cylinder 186 (COMPRESS-HI and 3344). The disk space allocated to the restored file is the same amount that was allocated to the original file. Also, all characteristics of the restored file except location and date are the same as the original file.

A parameter on the control statement enables the program to select any file on the tape. To locate the file, the parameter uses the file number that was assigned to the file during the backup function. If you omit this parameter, the program restores all the files that were copied during the backup function.

After a file is restored, the FILE xxxxxxxx HAS BEEN COPIED FROM POSITION zzzz ON TAPE message is issued.

After a successful restore function, the TAPE COPY IS COMPLETE message is issued.

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

Functions	Control Statements
Copy files from one main data area to another main data area, or move files within the same main data area	// COPYFILES FROM-code,TO-code [,PACKIN-name] [,PACKO-name] [,COMPRESS-{HI / LO}] // END

Back up a file(s) from disk to tape or restore a file(s) from tape to disk	// TAPEFILES {FROM / TO} -code[,LABEL-filename] [,PACK-name]
	[,SEQNUM-{1 / X / 'number'}] [,COMPRESS-{HI / LO}]
	// END

PARAMETER SUMMARY

Parameter	Description
COPYFILES Statement	
FROM-code	Specifies the location of the main data area that contains the files to be moved (compressed) or the files to be copied. Valid codes are those for the main data areas.
TO-code	Specifies the location of the main data area that contains the files to be moved (compressed) or receives the copied files. Valid codes are those for the main data areas.
PACKIN-name	Specifies the name of the main data area to be used as the input area (copy function) or to contain the moved (compressed) files (move function).
PACKO-name	Specifies the name of the main data area to be used as the output area (copy function) or to contain the compressed files (move function).
COMPRESS-LO	Allocates the first available space for the file starting from the beginning of the main data area (cylinder 1).
COMPRESS-HI	Allocates the first available space for the file starting from the end of the main data area (cylinder 166 for a 3340 or cylinder 186 for a 3344).

Parameter	Description
TAPEFILES Statement	
FROM-code	Specifies the location of the main data area that contains the file(s) to be backed up. Valid codes are those for the main data areas.
TO-code	Specifies the location of the main data area to which the files are to be restored. Valid codes are those for the main data areas.
LABEL-filename	Specifies the file that you choose to back up or restore. If this parameter is omitted, all files on the main data area or the tape are copied.
PACK-name	Specifies the name of the main data area that is used as input for a backup function or output for a restore function.
SEQNUM-1	Specifies that the first file is to be written on tape immediately following the volume label (backup function). Or, specifies that a file(s), starting with file number 1 is to be read during the restore function.
SEQNUM-X	If this parameter is specified on the first control statement, the program: – Writes file number 1 on tape immediately following the volume label (backup function); – Reads a file(s) starting with file number 1 during the restore function. When this parameter is included on any control statement other than the first one, the program increments the file number by 1 for each file processed.
SEQNUM-'number'	This parameter specifies the file number where file processing is to begin.
COMPRESS-LO	Allocates the first available space for the file starting from the beginning of the main data area (cylinder 1).
COMPRESS-HI	Allocates the first available space for the file starting from the end of the main data area (cylinder 166 for a 3340 or cylinder 186 for a 3344).

PARAMETER DESCRIPTION

FROM and TO Parameters (COPYFILES)

The FROM parameter (FROM-code) specifies the main data area that contains the files to be moved or copied. The TO parameter specifies the main data area that contains the files to be moved or receives the files specified in the FROM parameter. If the FROM and TO parameters are the same, the files on the specified main data area are moved. If the parameters are different, the files contained in the main data area specified by the FROM parameter are copied to the main data area specified by the TO parameter.

PACKIN and PACKO Parameters (COPYFILES)

These parameters are optional. They are used to verify that the correct data module is online and/or the correct main data area has been specified. Either one or both parameters may be supplied.

COMPRESS Parameter (COPYFILES)

This parameter determines the location that the file(s) will be moved or copied to relative to the beginning (cylinder 1) or the end (cylinder 166 for a 3340 or cylinder 186 for a 3344) of the main data area. The COMPRESS-LO parameter causes the file(s) to be moved to the available space at the beginning of the main data area. The COMPRESS-HI parameter causes the file(s) to be moved to the available space at the end of the main data area.

FROM Parameter (TAPEFILES)

The FROM parameter (FROM-code) serves a dual purpose. It tells the program to perform the backup function. It also specifies the main data area that contains the files to be backed up.

TO Parameter (TAPEFILES)

The TO parameter (TO-code) serves a dual purpose. It tells the program to perform the restore function. It also specifies the main data area that is to receive the restored files.

LABEL Parameter (TAPEFILES)

This optional parameter specifies the name of the file that is to be backed up or restored. If the program performs the backup function and this parameter is omitted, all files on the main data area specified in the FROM parameter are copied to tape (backed up).

If the program performs the restore function and this parameter is omitted, the program copies or attempts to copy all files from the tape to the main data area specified in the TO parameter (restore). If the program cannot find sufficient disk space for the files, a message is issued.

PACK Parameter (TAPEFILES)

This optional parameter specifies the name of the main data area that either contains the input files for the backup function or receives the output files for the restore function.

SEQNUM Parameters (TAPEFILES)

During the backup function, this optional parameter (SEQNUM-1) specifies that the first file processed for the control statement is to be assigned file number 1 and written to the first location on tape. During the restore function, the first file that the program reads is file number 1.

The optional parameter SEQNUM-X causes the program to operate differently depending on when the control statement containing the parameter is processed. If you specify this parameter on a single control statement or the first control statement of a group, the program operation is the same as when you specify the parameter SEQNUM-1. If you specify SEQNUM-X on any control statement except the first one, the program increments by 1 the file number that was assigned to the last file processed and assigns the updated number to the first file processed for this control statement.

During the backup function, the optional parameter SEQNUM-'number' allows you to assign the first file number and the location to the first file processed for the control statement.

During the restore function, the SEQNUM-'number' parameter allows you to select the starting location of the file(s) to be restored.

COMPRESS Parameter (TAPEFILES)

The COMPRESS parameter is ignored for the backup function. During the restore function, this parameter specifies where the file is to be located relative to the beginning and the ending cylinders of the main data area. If COMPRESS-LO is specified, the file(s) will be located in the first available space starting at cylinder 1; if COMPRESS-HI is specified, the file(s) will be located in the first available space toward the end cylinder of the main data area.

CONSIDERATIONS AND RESTRICTIONS

- File compress program functions are effective only for main data areas.
- Unexpected messages and an unusable file can result from canceling the partition that is executing the file compress program.
- A file with the same name and date as a file in the TO area is not copied.
- An indexed multivolume file is not copied if two indexed multivolume files already exist in the TO area.
- A multivolume file with the same name as an existing file in the TO area is not copied.
- A file with the same name as an existing multivolume file in the TO area is not copied.
- The LOCATION parameter on the FILE statement must be changed after the file is moved.
- A file is not copied when space is not available in the TO area.
- File compress program cannot execute if spool is active on the pack specified in the TO parameter. Stop spool until $FCOMP has finished.
- The location of the space that is allocated for an added file(s) is determined by the smallest available space that will contain the file(s). It is possible for an added file to be located near the low end of the main data area even though COMPRESS-HI is specified. Likewise, an added file can be located at the high end of the main data area even though COMPRESS-LO is specified.
- A $SPOOL, $CCPDUMP, or $MONITOR file is not copied if encountered during the copy, move, or backup function.
- All main data area codes are valid for the TO and FROM parameters on the COPYFILES or TAPEFILES statement.
- $FCOMP attempts to copy all files (regardless of their size), one file at a time, from one main data area to another main data area. If a file is encountered on the FROM main data area that will not fit into the available storage space on the TO main data area, then $FCOMP issues a message and halts. The message identifies the file that did not copy and informs the operator of the available options. This situation applies to all files, regardless of their location (cylinders 1 through 166 on a 3340, or cylinders 1 through 186 on a 3344).
- Only the actual records of a file are copied to tape (backed up). When the last record of a file is copied, the program considers the file completely copied, even though there may be more space allocated to the file.
- When the backup function is used to copy one or more disk files to tape or the restore function is used to copy one or more tape files to disk, the entire main data area on the disk is dedicated to the partition. If any files are allocated on the main data area, the partition will issue a message.

FILE STATEMENT CONSIDERATIONS AND RESTRICTIONS (BACKUP AND RESTORE)

- Conversion must be specified when 7-track tape is used.
- The record format (RECFM) entry defaults to fixed (F) even if another format is specified.
- If an END parameter is not specified, the default is UNLOAD.
- BACKUP must be specified as the NAME parameter.
- The record length is always equal to the block length. The valid block lengths are:

1536	1/8-track transfer (default value)
3072	1/4-track transfer
6144	1/2-track transfer
12288	1-track transfer
24576	2-track transfer

- A labeled tape must be used to ensure that the tape file sequence can be displayed at a later time. (A REEL parameter must be specified on the FILE statement.)
- The SEQNUM keyword defaults to the value specified on the TAPEFILES statement.
- The label of the tape for the file is the same as the header label.
- Multivolume tape files are supported.

OCL CONSIDERATIONS

The following OCL statements are needed to load the file compress program:

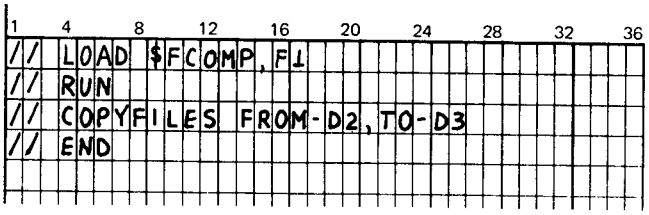
```
// LOAD $FCOMP,code
// RUN
```

The code you supply depends on the location of the simulation area containing the program. Possible codes are R1, F1, R2, F2.

EXAMPLES

Figures 4-49, 4-50, 4-51, and 4-52 show the OCL and control statements to copy and move files.

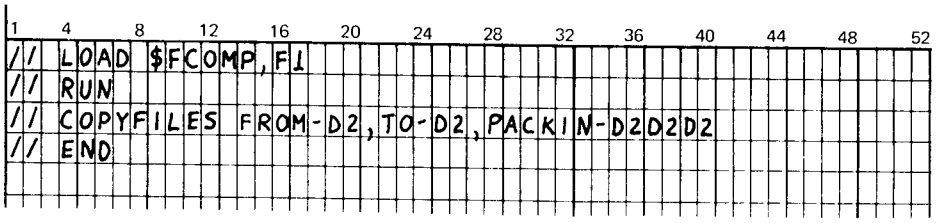
Figures 4-53 through 4-59 show the OCL and control statements to back up and restore files.



Explanation:

- The file compress program is loaded from F1.
- Control statement explanation:
  - The files on the main data area on drive 2 (FROM-D2) are added (copied) to the existing files on the main data area D3 (TO-D3).

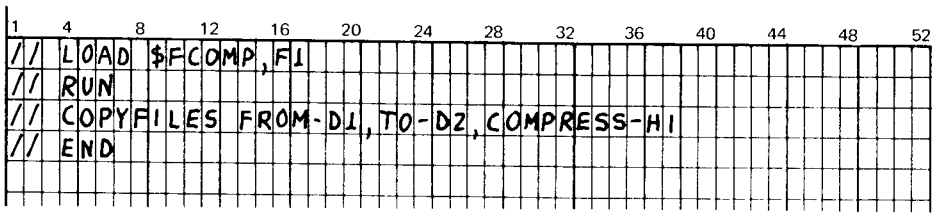
Figure 4-49. OCL and Control Statements for Copying Files



Explanation:

- The file compress program is loaded from F1.
- Control statement explanation:
  - The files on the main data area on drive 2 are moved (FROM-D2,TO-D2).
  - The name of the main data area on drive 2 is verified (PACKIN-D2D2D2).

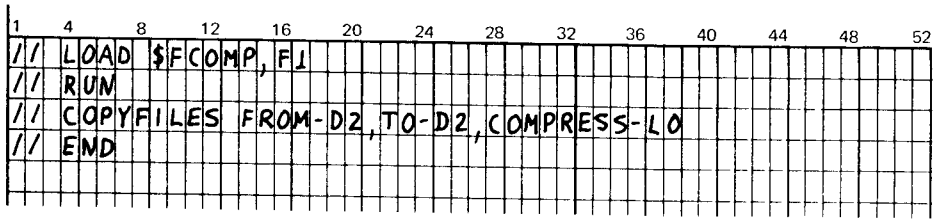
Figure 4-50. OCL and Control Statements for Moving Files



Explanation:

- The file compress program is loaded from F1.
- Control statement explanation:
  - The files on the main data area on drive 1 (FROM-D1) are added (copied) to the existing files on the main data area on drive 2 (TO-D2).
  - The files are added to the high end of the main data area on D2 (COMPRESS-HI).

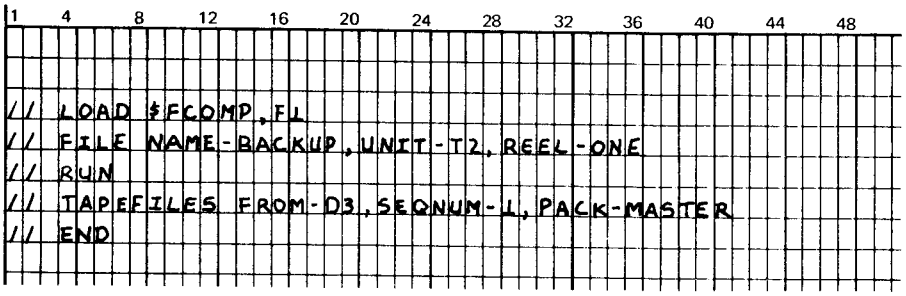
Figure 4-51. Copy Files from a Main Data Area to the High End of Another Main Data Area



Explanation:

- The file compress program is loaded from F1.
- Control statement explanation:
  - The files on the main data area on D2 are moved toward the beginning of the main data area (COMPRESS-LO).

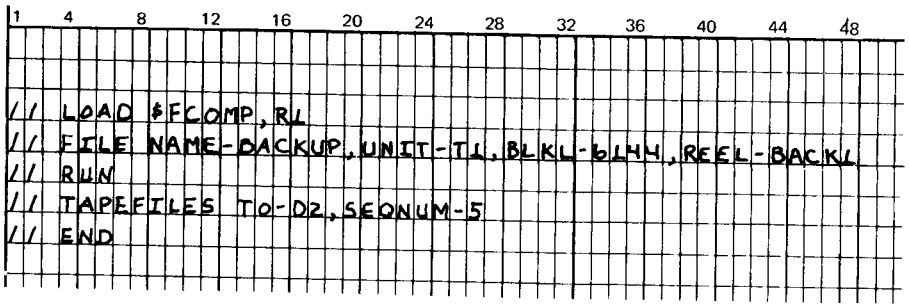
Figure 4-52. Move Files toward the Beginning of the Main Data Area



Explanation:

- The file compress program is loaded from F1.
- Output file (OCL sequence):
  - The file name is always BACKUP (NAME-BACKUP).
  - The copy will go to tape unit 2 (UNIT-T2).
  - The tape unit is a 9-track drive.
  - The tape volume label is ONE (REEL-ONE).
- Control statement explanation:
  - The files are located on drive 3 (FROM-D3).
  - The first files are copied to the first position on tape (SEQNUM-1).
  - The pack name on drive 3 must be MASTER (PACK-MASTER).

Figure 4-53. OCL and Control Statements to Back Up All the Files on a Main Data Area to Tape



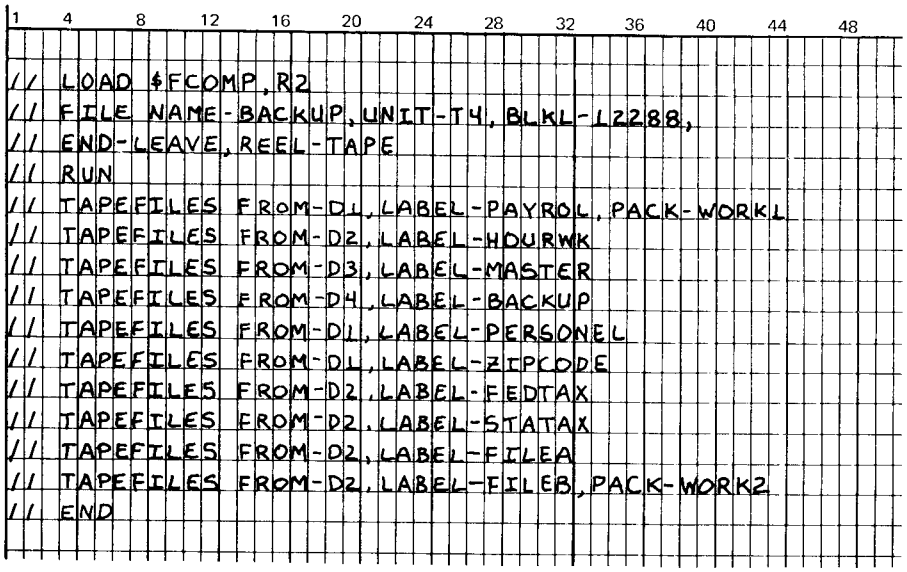
Explanation:

- The file compress program is loaded from R1.

- Input file (OCL sequence):
  - The file name is always BACKUP (NAME-BACKUP).
  - The backup files are contained on tape drive 1 (UNIT-T1).
  - The label of the tape volume is BACK1 (REEL-BACK1).
  - Tape unit 1 is a 9-track drive.
  - Block and record lengths are 6144 (BLKL-6144).

- Control statement explanation:
  - The backup files are copied to drive 2 (TO-D2).
  - The files are copied starting at the fifth position (SEQNUM-5) on the tape.

Figure 4-54. OCL and Control Statements to Restore the Files from a Specific File Number



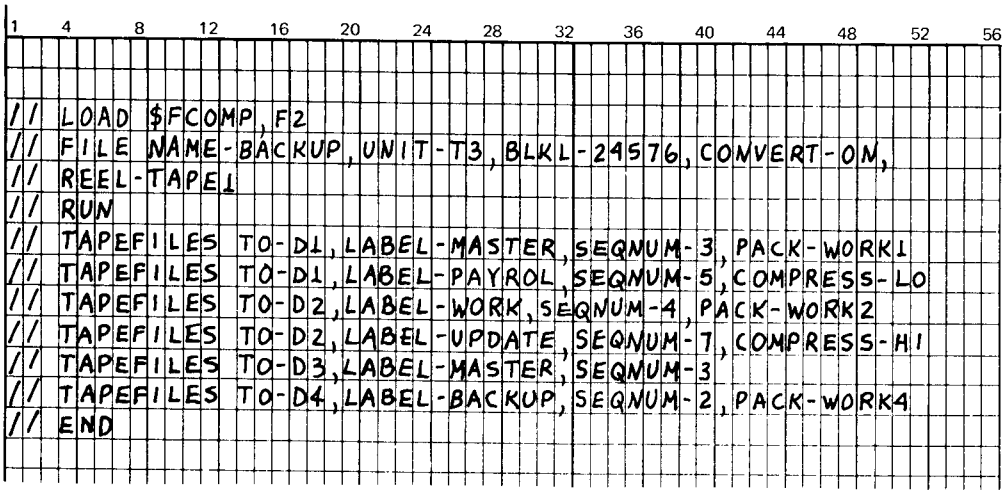
Explanation:

The file compress program is loaded from R2.

- Output file (OCL sequence):
  - The file name is always BACKUP (NAME-BACKUP).
  - The files are copied to tape unit 4 (UNIT-T4).
  - The tape unit is a 9-track drive.
  - Block and record lengths are 12288 (BLKL-12288).
  - The label of the tape volume is TAPE (REEL-TAPE).
  - The tape remains in the position it was in after the last record is written (END-LEAVE).

- Control statement explanation:
  - The file named PAYROL (LABEL-PAYROL) is copied from drive 1 (FROM-D1) to the first position on tape (the default of the SEQNUM parameter is position 1).
  - The file named HOURWK (LABEL-HOURWK) is copied from drive 2 (FROM-D2) to the second position on tape.
  - The file named MASTER (LABEL-MASTER) is copied from drive 3 (FROM-D3) to the third position on tape.
  - The file named BACKUP (LABEL-BACKUP) is copied from drive 4 (FROM-D4) to the fourth position on tape.
  - The files named PERSONEL and ZIPCODE (LABEL-PERSONEL, LABEL-ZIPCODE) are copied from drive 1 (FROM-D1) to the fifth and sixth positions.
  - The files named FEDTAX, STATAX, FILEA, and FILEB are copied from drive 2 (FROM-D2) to the seventh, eighth, ninth, and tenth positions.
  - The pack name is WORK1 on drive 1.
  - The pack name is WORK2 on drive 2.

Figure 4-55. OCL and Control Statements to Back Up Selected Files



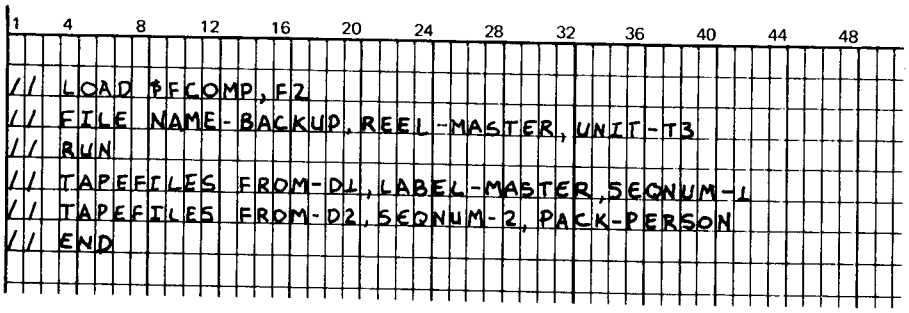
Explanation:

- The file compress program is loaded from F2.

- Input file (OCL sequence):
  - The file name is always BACKUP (NAME-BACKUP).
  - Tape unit 3 contains the backup files (UNIT-T3).
  - The label of the tape volume is TAPE1 (REEL-TAPE1).
  - Block and record lengths are 24576 (BLKL-24576).
  - Tape unit 3 is a 7-track drive (CONVERT-ON).
  - CONVERT-ON indicates data conversion.

- Control statement explanation:
  - The file named MASTER (LABEL-MASTER) is copied to drive 1 (TO-D1) from the third position on tape (SEQNUM-3).
  - The pack name must be WORK1 on drive 1 (PACK-WORK1).
  - The file named PAYROL (LABEL-PAYROL) is copied to the low end (COMPRESS-LO) of the main data area on drive 1 (TO-D1) from the fifth position on tape (SEQNUM-5).
  - The file named WORK (LABEL-WORK) is copied to drive 2 (TO-D2) from the fourth position on tape (SEQNUM-4).
  - The pack name must be WORK2 on drive 2 (PACK-WORK2).
  - The file named UPDATE (LABEL-UPDATE) is copied to the high end (COMPRESS-HI) of the main data area on drive 2 (TO-D2) from the seventh position on tape (SEQNUM-7).
  - The file named MASTER (LABEL-MASTER) is copied to drive 3 (TO-D3) from the third position on tape (SEQNUM-3).
  - The file named BACKUP (LABEL-BACKUP) is copied to drive 4 (TO-D4) from the second position on the tape (SEQNUM-2).
  - The pack name must be WORK4 on drive 4 (PACK-WORK4).

Figure 4-56. OCL and Control Statements to Restore Selected Files



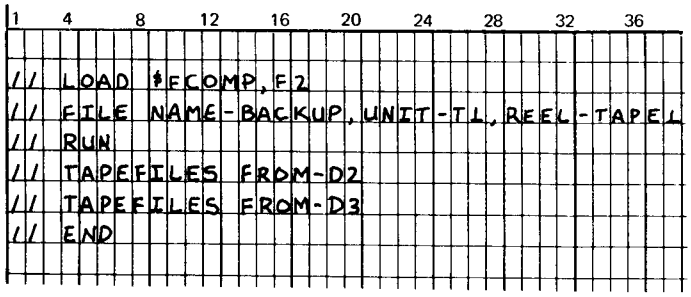
Explanation:

- The file compress program is loaded from F2.

- Output file (OCL sequence):
  - The file name is always BACKUP (NAME-BACKUP).
  - The backup files are copied to tape unit 3 (UNIT-T3).
  - The tape unit is a 9-track drive.
  - The block and record lengths are 1536 (default value).
  - The label of the tape volume is MASTER (REEL-MASTER).

- Control statement explanation:
  - The file named MASTER (LABEL-MASTER) is copied to position 1 (SEQNUM-1) on the tape.
  - The files located on drive 2 are copied to the tape starting at position 2 (SEQNUM-2).
  - The pack name on drive 2 must be PERSON (PACK-PERSON).

Figure 4-57. OCL and Control Statements to Back Up One File from a Main Data Area and All the Files from Another Main Data Area



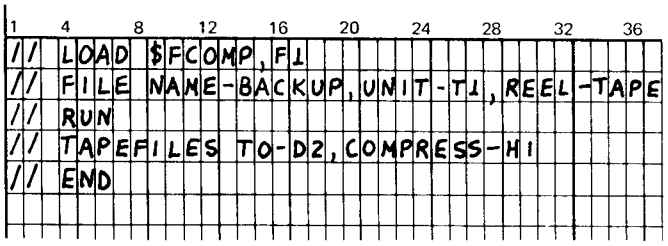
Explanation:

- The file compress program is loaded from F2.

- Output file (OCL sequence):
  - The file name is always BACKUP (NAME-BACKUP).
  - The files are copied to tape unit 1 (UNIT-T1).
  - The tape unit is a 9-track drive.
  - The block and record lengths are 1536 (defaulted).
  - The label of the tape volume is TAPE1 (REEL-TAPE1).

- Control statement explanation:
  - The files located on drive 2 (FROM-D2) are copied to tape starting at position 1.
  - The files located in drive 3 (FROM-D3) are copied to tape starting at the next available position.

Figure 4-58. OCL and Control Statements to Back Up All the Files Contained on Two Main Data Areas



Explanation:

- The file compress program is loaded from F1.

- Input file (OCL sequence):
  - The file name is always BACKUP (NAME-BACKUP).
  - The backup files are located on T1 (UNIT-T1).
  - The label of the tape volume is TAPE (REEL-TAPE).

- Control statement explanation:
  - The backup files are copied to drive 2 (TO-D2).
  - The copied files are located at the high end of the main data area (COMPRESS-HI).

Figure 4-59. Restore Backup Files to the High End of the Main Data Area

System History Area Display Program—$HIST

PROGRAM DESCRIPTION

The system history area display program has the following function:

Print and/or copy to a device independent file a portion of or the entire system history area (SHA).

The SHA is a special area on the system pack (the pack from which an IPL was performed) that contains the following information:

- OCL statements read by the system
- OCL diagnostics issued
- Control statements for system service programs read by the system
- OCC entered by the operator
- OCC diagnostics issued
- System messages issued
- Unit record restart messages
- Operator responses to system messages
- Display screen images

The output of $HIST is an audit trail of system activity that can be used to debug program problems. This output will be copied to a device independent file, for use by other audit programs, by supplying a $HISTORY FILE statement (NAME-$HISTORY). SHA records written to unit record devices are truncated to the unit record length.

The SHA is printed if you supply a PRINT control statement. The SHA is copied, not printed, if you supply an OUTPUT control statement. However, you must also supply a FILE statement with the OUTPUT control statement. The name of the file must be $HISTORY.

You may choose not to supply a control statement. If so, $HIST will perform the functions of the following control statement and parameter:

```
// PRINT HISTORY-CURRENT
```

Note: The system history area copy program ($HACCP) can be used to copy the current portion of the SHA to a disk file. This program can be automatically invoked if the communications control program (CCP) is executing. For information about the system history area copy program ($HACCP), see the *IBM System/3 Communications Control Program System Reference Manual,* GC21-7620.

CONTROL STATEMENT SUMMARY

The control statements you supply depend on the desired results.

Functions	Control Statement
Print SHA entries	// PRINT HISTORY-{ALL / CURRENT}
Copy SHA entries to a device independent file	// OUTPUT HISTORY-{ALL / CURRENT}

PARAMETER SUMMARY

Parameter	Description
HISTORY-ALL	Process all entries in SHA
HISTORY-CURRENT	Process SHA entries not previously printed and/or copied

PARAMETER DESCRIPTIONS

HISTORY Parameter

The HISTORY parameter allows you to select which part of the SHA that you wish to print and/or copy. The entire SHA is processed if HISTORY-ALL is specified. If HISTORY-CURRENT is specified, only those entries not previously printed and/or copied are processed. If the PRINT or OUTPUT control statement is omitted, HISTORY-CURRENT is assumed.

OCL CONSIDERATIONS

The following OCL statements are needed to load the system history area display program:

```
// LOAD $HIST,code
// RUN
```

The code you supply depends on the location of the simulation area containing the program. Possible codes are R1, F1, R2, F2.

The following OCL statements are needed to load the system history area display program and copy the output to a device supported by device-independent data management:

```
// LOAD $HIST,code
// FILE NAME-$HISTORY,UNIT-code, . . .
// RUN
```

Figure 4-60 is a sample printout of the system history area. For additional information regarding the entries in the system history area, see the appropriate related publication listed in the *Preface.*

```
$HIST    PRINT SYSTEM HISTORY AREA - ALL       PAGE 013    DATE 12/31/75  TIME-00.04.55

(1) 1 * 1           R008         IPL
(2)ENTER SYSTEM DATE
    $ - $ ID OD D1 I
(3)     DATA MODULE D1D1D1 IS ON D1
    $ - $ ID OD D2 I
        DATA MODULE D2D2D2 IS ON D2
    $ - $ ID OD D3 I
        DATA MODULE D3D3D3 IS ON D3
    $ - $ ID OD D4 I
        DATA MODULE D4D4D4 IS ON D4
(4) 1 * 1           R008         IPL
    12/31/75
    3 - 3 CT EJ     D        F1 D1A  IPLED
        PROGRAM END
(5) 2 - 2 CT EJ     D        F1 D1A  IPLED
        PROGRAM END
    1 - 1 CT EJ     D        F1 D1A  IPLED
(6)     PROGRAM END
(7)TM 102738
(7)S SP
    1 - 1 SP UT FA I
        SPOOL FILE STARTED - NEW,D4,50,4
    1 - 1 SP UT SA I
(8)     SPOOL IS ACTIVE P1
    1 - 1 SP UT SA I
        SPOOL IS ACTIVE P2
    1 - 1 SP UT SA I
        SPOOL IS ACTIVE P3
(9)S R
(10) S - S IF CB    D  123
        1442 NOT READY
(11)S 1 S 1442 RESTARTED
(10) S - S IF CB    D  123
        1442 NOT READY
(12)D RQ
(11) S 1 S 1442 RESTARTED
(13)CN *RDRQ
(8)  S - S SP UT RQ I
        RDRQ IS CANCELED
(12)D RQ
(8)  S - S SP UT RT I
        SPOOL RDR TERMINATED
(14)RDR P1,1442
(15) 1 - 1 CT EJ    D        F1 D1A  IPLED
    1 //MAINT JOB PARTITION-1
    1 // LOAD $MAINT,F1
(16) 1 // RUN
    1 // COPY FROM-READER,TO-F1,RETAIN-R,LIBRARY-0,NAME-$TRACE
    1 // END
    1 - 1 CT ES     I       MAINT     $MAINT01
        12/31/75 10.31.19 10.31.25
(17) 1 /.
    1 - 1 CT EJ     I       MAINT     STEP
        12/31/75 10.31.19 10.31.26
    1 //HIST JOB PARTITION-1
    1 // CALL AB,F1
(18) 1 XXDH LOAD $HIST,F1
    1 XX RUN
    1 // RUN
(19)D H
(20) 1 // PRINT HISTORY-CURRENT
    1 // END
    1 - 1 CT ES     I       HIST      DH
        12/31/75 10.31.26 10.31.37
(21) 1 /.
    1 - 1 CT EJ     I       HIST      STEP
        12/31/75 10.31.26 10.31.38
(22) S - S SP UT CF D 01    HIST      DH
        CHANGE FORM TO TYPE BBB
(23) 1 - 1 SP UT RE D  1 3 JOB        STEP
        RDRQ IS EMPTY/HELD
(24)S 1 S SP UT CF D 01    HIST      DH
(25) S - S SP UT TM I       HIST      DH
        12/31/75 10.32.02 10.32.53 PRT
    2 - 2 CT EJ     D        F1 D1A  IPLED
    2 //A  JOB
(26) 2 // LOAD $HIST,F1
    2 // RUN
    2 // PRINT HISTORY-ALL
    2 // END
```

Figure 4-60 (Part 1 of 2). Printout of System History Area

(1) This message is a result of pressing the PROGRAM LOAD key.

(2) Date prompt.

(3) This system information message indicates which 3340 data modules are online at IPL time.

(4) Operator response to the date prompt.

(5) End of job message for each partition indicates IPL is complete.

(6) Command to set time of day. Format is HH.MM.SS.

(7) Command to start spool.

(8) Spool status informational messages.

(9) Command to start spool reader.

(10) System message caused by a 1442 not ready condition.

(11) Message generated as a result of unit record restart.

(12) Command to display reader queue.

(13) Command to cancel reader queue.

(14) Command to assign the 1442 as the system input device for partition 1.

(15) Partition 1 is started.

(16) OCL and control statements to load and execute a program.

(17) End of step (ES) and end of job (EJ) messages. These messages are informational (I). They do not require a response. Job name is MAINT. Program name is $MAINT. Step number is 01. The job step was executed on 12/31/75. The time of day when program started executing was 10.31.19; the program ended execution at 10.31.25.

(18) OCL statements to call a procedure (AB) which loads and executes $HIST.

(19) Command to display the system history area on the display screen.

(20) Control statements for $HIST.

(21) System messages are similar to (17). Program name is replaced by stepname (DH).

(22) System message that was issued by spool informing the operator to change forms.

(23) System message informing the operator that the input queue is empty/held.

(24) Operator response to (22) with a 1 option.

(25) SPOOL time recording optional message indicating time required to print (PRT) the output of jobname HIST, stepname DH.

(26) OCL and control statements used to obtain this sample printout of the system history area.

Figure 4-60 (Part 2 of 2). Printout of System History Area

EXAMPLES

Figure 4-61 shows the OCL statement to load the system history area display program. Figures 4-62 through 4-66 show the OCL and control statements to print and/or copy the SHA.

```
1   4   8   12  16  20  24  28  32  36
/E
// LOAD $HIST,F1
// RUN
```

Explanation:

- The system history area display program is loaded from F1.

Figure 4-61. OCL Statement to Load the System History Area Display Program

```
1   4   8   12  16  20  24  28  32  36
// PRINT HISTORY-ALL
// END
```

Explanation:

- The output is printed (PRINT).
- All entries are printed (HISTORY-ALL).

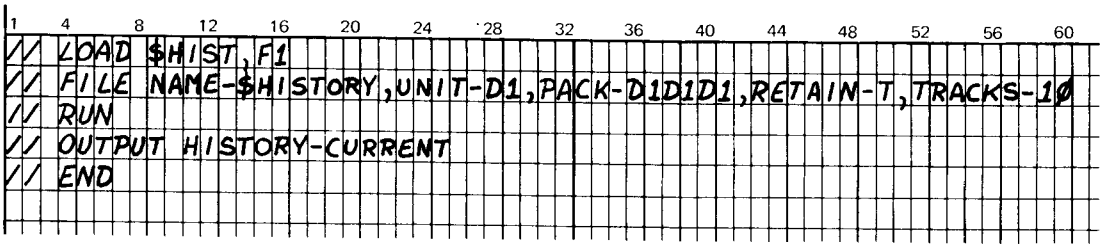
Figure 4-62. Control Statements to Print the Entire Contents of the System History Area

```
1   4   8   12  16  20  24  28  32  36
// LOAD $HIST,F1
// FILE UNIT-T1,NAME-$HISTORY,REEL-NL
// RUN
// PRINT HISTORY-ALL
// END
```

Explanation:

- The system history area display program is loaded from F1.
- The SHA is copied to tape (UNIT-T1).
- The SHA is printed (PRINT statement).
- All entries of the SHA are copied and printed (HISTORY-ALL).

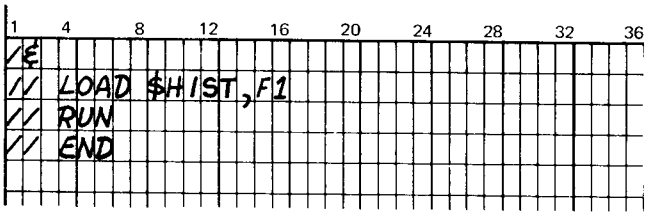
Figure 4-63. OCL and Control Statements to Copy and Print All Entries in the System History Area



Explanation:

- The system history area display program is loaded from F1.
- The FILE statement causes the output to be copied to a disk file on D1 (UNIT-D1).
- Output is not printed (OUTPUT statement); only the current entries are copied to the disk file (HISTORY-CURRENT).

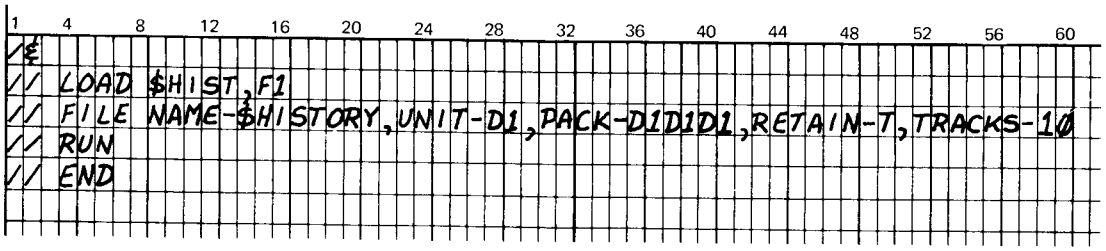
Figure 4-64. OCL and Control Statements to Copy the Current Entries to a Disk File Without Printing



Explanation:

- The system history area display program is loaded from F1.
- The current entries in the SHA are printed. (A control statement and parameter is not supplied.)

Figure 4-65. OCL Statements to Print the Current Entries in the System History Area



Explanation:

- The system history area display program is loaded from F1.
- Since a control statement is not supplied, the $HIST program function defaults to the following control statement:

  // PRINT HISTORY-CURRENT

- The current entries in the SHA are copied to a disk file. The name of the file is $HISTORY (NAME-$HISTORY). The file is on D1 (UNIT-D1).
- The current entries in the SHA are printed.

Figure 4-66. OCL Statements to Print and Copy the Current Entries in the System History Area

Disk Initialization Program—$INIT

PROGRAM DESCRIPTION

The disk initialization program has the following functions:

- Write track and record addresses.
- Check for defective tracks (a process called surface analysis).
- Assign alternate tracks to any defective tracks.
- Write a volume label to identify each volume.
- Format the volume table of contents (main data area only).

Before executing this program, you should be familiar with the volume format described in Part 3 of this manual.

All volumes must be initialized before use. Volumes that have been initialized need not be reinitialized unless you want to erase their contents and rename them.

This program can initialize a maximum of five volumes during one execution.

TYPES OF INITIALIZATION

There are five types of initialization:

- FORCE is used primarily to initialize a new volume. This type of initialization erases all files on the entire volume (a main data area and its associated simulation areas).
- CLEAR initializes a main data area without checking for active files. The area must have been previously initialized.
- PRIMARY initializes a main data area if there are no active files in that area.
- CYL0 initializes only cylinder 0 of a main data area. This is a fast way to reinitialize a main data area that had been previously initialized.
- RENAME is used to change the name, ID, and NAME360 of a main data area without affecting the contents of that area. The area must have been previously initialized.

Note: Except for FORCE, all types of initialization apply only to the main data area. The simulation area program, $SCOPY, is used for other functions associated with the simulation areas.

CAUTION
CLEAR and FORCE erase any active files on the area(s). CLEAR erases files only in the main data area. FORCE erases files on the entire volume (a main data area and its associated simulation areas). CYL0 destroys any current VTOC entries in the main data area.

CONTROL STATEMENT SUMMARY

The control statement you must supply depends on the desired results.

Type of Initialization	Control Statements (1)

FORCE (2)	// UIN TYPE-FORCE (3),UNIT-{code / 'codes'} [,ERASE-{NO / YES}]
	// VOL PACK-name[,ID-characters] [,NAME360-characters]
	// END

PRIMARY (2)	// UIN TYPE-PRIMARY,UNIT-{code / 'codes'} [,VERIFY-number]
	// VOL PACK-name[,ID-characters] [,NAME360-characters] [,OLDPACK-name]
	// END

CLEAR (2)	// UIN TYPE-CLEAR,UNIT-{code / 'codes'} [,VERIFY-number]
	// VOL PACK-name[,ID-characters] [,NAME360-characters] [,OLDPACK-name]
	// END

CYL0 (2)	// UIN TYPE-CYL0,UNIT-{code / 'codes'}
	// VOL PACK-name[,ID-characters] [,NAME360-characters]
	[,OLDPACK-characters]
	// END

RENAME (2)	// UIN TYPE-RENAME,UNIT-{code / 'codes'}
	// VOL PACK-name[,ID-characters] [,NAME360-characters]
	[,OLDPACK-characters]
	// END

Volume previously initialized

Note: The control statement defaults to TYPE-FORCE if the data module is still in System/370 format and TYPE-CLEAR or PRIMARY initialization has been specified. If CYL0 or RENAME is specified and the data module is still in System/370 format, the system issues an error message.

(1) Control statements are required in the order they are listed: UIN, VOL, END.

(2) One VOL statement is required for each main data area listed in the UNIT parameter of the UIN statement. The PACK parameter in the first VOL statement applies to the first main data area disk listed in the UNIT parameter. The PACK parameter in the second VOL statement applies to the second main data area listed in the UNIT parameter.

(3) If the TYPE parameter is omitted, TYPE-PRIMARY is assumed.

PARAMETER SUMMARY

Parameter	Description
UIN (Input Definition) Statement	
TYPE-FORCE	If the TYPE parameter FORCE is used, a main data area and its associated simulation areas are initialized without a check for active files.
TYPE-PRIMARY	Primary initialization (main data area only). Tracks already initialized are reinitialized. The program does not initialize disks containing temporary or permanent data files.
TYPE-CLEAR	Clear initialization (main data area only). Tracks already initialized are reinitialized. Active file checking is bypassed and any data on the tracks is destroyed.
TYPE-CYL0	CYL0 is an abbreviated initialization, initializing only cylinder 0 on a main data area that has been previously initialized on a System/3. This includes rewriting the volume label (PACK), the ID, and NAME360 entries, and deleting any VTOC entries that may be present on the main data area.
TYPE-RENAME	RENAME initialization applies only to those entries on cylinder 0 of a main data area that contain the volume label (PACK), ID, and NAME360. Default values for ID and NAME360 are used if the parameters are not supplied.
UNIT-code	One main data area. Possible codes are those for the main data areas.
UNIT-'code,code'	Multiple main data areas (maximum of five). Possible codes are those for the main data areas.
VERIFY-number	Surface analysis. Done the number of times indicated (number can be 1-255). VERIFY-16 is assumed if the parameter is omitted. This parameter is only used for TYPE-CLEAR and TYPE-PRIMARY initialization.
ERASE-{NO / YES}	TYPE-FORCE initialization only. If you do not want to retest defective tracks use ERASE-NO. If you want to retest defective tracks, use ERASE-YES.
VOL (Volume) Statement	
PACK-name	Name of the main data area. Can contain any of the standard System/3 characters except apostrophes, leading or embedded blanks, and embedded commas (1). Its length must not exceed 6 characters.
ID-characters	Additional identification. Can contain any of the standard System/3 characters except apostrophes, leading or embedded blanks, and embedded commas (1). Its length must not exceed 10 characters. If you omit this parameter, blanks are written in the ID field.

Parameter	Description
NAME360-characters	Additional identification for the main data area. The name will be placed in the System/360 format 1 DSCB. Can contain any of the standard System/3 characters except apostrophes, leading or embedded blanks, and embedded commas (1). Its length must not exceed 44 characters. If you omit this parameter, the program defaults to SYSTEM/3.DATA.
OLDPACK-name	Current name of the main data area to be initialized. Name can be any of the standard System/3 characters except apostrophes, leading or embedded blanks, and embedded commas (1). Its length must not exceed 6 characters.

(1) This is due to their delimiter function.

PARAMETER DESCRIPTIONS

TYPE Parameter (UIN)

The TYPE parameter indicates the type of initialization you want to do: PRIMARY, FORCE, CLEAR, CYL0, or RENAME. The type of initialization determines which tracks will be initialized.

PRIMARY Initialization

PRIMARY initialization applies to main data areas you have used but want to initialize again. Tracks that were previously initialized are initialized again. Any data on the tracks is destroyed. You can use PRIMARY initialization as often as you want. However, the program will not initialize main data areas containing temporary or permanent data files. You must delete the files using the file delete program $DELET.

FORCE Initialization

FORCE initialization applies to volumes that are not formatted for System/3. FORCE may also be used to reinitialize an entire volume that you have used.

Note: The simulation area program, $SCOPY, must be used after a FORCE initialization to reformat the simulation areas.

CLEAR Initialization

The results of a CLEAR initialization are functionally equivalent to PRIMARY except that CLEAR initializes main data areas containing temporary or permanent data files.

CAUTION
All temporary data files or permanent data files are completely erased.

CYL0 Initialization

Cylinder zero (CYL0) initialization can be used if you want to reinitialize only cylinder 0.

RENAME Initialization

RENAME initialization may be used if you want to change PACK, ID, and NAME360 parameters. Default values are used if the ID and NAME360 parameters are not supplied.

Note: If an invalid System/3 label is found during RENAME initialization, a reinitialization must be done with FORCE, CLEAR, PRIMARY, or CYL0.

UNIT Parameter

The UNIT parameter (UNIT-code) indicates the location of the volume you want to initialize. The program can initialize up to five volumes during one program run.

The form of the UNIT parameter depends on the number of volumes you are initializing:

- For one volume, use UNIT-code.
- For two volumes, use UNIT-'code,code'.

Possible codes are those for the main data area.

For all initialization, the order of codes must correspond to the order of VOL control statements. If, for example, you had used the parameter UNIT-'D1,D2', the first VOL statement applies to the data module on drive 1, and the second to the data module on drive 2.

VERIFY Parameter

The VERIFY parameter (VERIFY-number), used only for CLEAR and PRIMARY initialization concerns surface analysis. It enables you to indicate the number of times you want the program to do surface analysis on suspected defective tracks before judging whether or not tracks are defective. The number can be from 1 to 255.

ERASE Parameter (UIN)

The ERASE parameter concerns alternate track assignment. It applies only to volumes that have already been initialized and used, but which you are reinitializing using TYPE-FORCE initialization.

The condition of tracks on such volumes was tested at least once before (during the previous initialization) and tracks that were found to be defective during surface analysis were assigned alternates. The ERASE parameter, therefore, enables you to indicate whether you want the program to (1) retest the tracks to which alternate tracks are already assigned or (2) leave the alternate tracks assigned without retesting the tracks.

The parameter ERASE-YES means to retest. If you tell the program to retest, it erases any existing alternate track assignments and tests all tracks.

The parameter ERASE-NO means not to retest. If you tell the program not to retest, it tests only those tracks to which no alternate tracks are assigned. Alternate tracks previously assigned remain assigned.

Defective tracks are not retested if the ERASE parameter is omitted.

Surface Analysis

Surface analysis is a procedure for testing the condition of tracks. It consists of writing test data on tracks, then reading the data to ensure that it was recorded properly.

In judging whether or not tracks are defective, the program does surface analysis the number of times you specify in the VERIFY parameter. If you omit the VERIFY parameter, surface analysis is done 16 times. Tracks that cause reading or writing errors any time during surface analysis are considered defective. Defective tracks can be assigned alternates. Each volume has 40 alternate tracks available. If the program finds more than 40 defective tracks, it considers the disk unusable and stops initializing it.

Alternate Track Assignment

Alternate track assignment is the process of assigning an alternate track to a defective track. If the disk initialization program finds a defective track during execution it assigns an alternate track to the defective track. The alternate is, in effect, a substitute for the defective track. Any time a program attempts to use the defective track, it automatically uses the alternate instead. Each volume has 40 alternate tracks.

If tracks become defective after a volume is initialized, another program (see *Alternate Track Assignment Program*) is used to assign alternate tracks. Volumes need not be reinitialized to assign alternate tracks.

Note: During a CLEAR or PRIMARY initialization, suspected defective tracks may be encountered within an active or IPL simulation area for the volume being initialized. If so, a message is issued prior to end of job that suggests $ALT be run against the volume after the conditions that caused the message have been eliminated.

PACK Parameter (VOL)

The PACK parameter (PACK-name) applies to all types of initialization. During initialization, the disk initialization program writes a volume label on each main data area. It uses the name you supply in the corresponding PACK parameter. (One VOL control statement containing a PACK parameter is required for each unit.)

The name can be any combination of standard System/3 characters except apostrophes, leading or embedded blanks, and embedded commas (because of their delimiter function). (See Appendix A for a list of standard System/3 characters.) Its length must not exceed 6 characters. Examples of valid volume label names are 0,F0001, 012, A1B9, and ABC.

In general, volume names are used for checking. Before a program uses a volume, its name is compared with a name you supply (either in OCL statements or control statements required by the program). If the names do not match, the program halts and prints a message. In this way, programs cannot use the wrong volumes without the operator knowing about it.

ID (Identification) Parameter (VOL)

The ID parameter (ID-characters) applies to all types of initialization. It enables you to include a maximum of 10 characters, in addition to the volume label name, to further identify a volume. The characters can be any combination of standard System/3 characters (Appendix A) except apostrophes, leading or embedded blanks, and embedded commas (because of their delimiter function). The information is strictly for your use; the system does not use it for checking. If you use the file and volume label display program to print the name, that program will also print the additional identification for you.

NAME360 Parameter (VOL)

The NAME360 parameter (NAME360-name) is used to specify a filename for data interchange with System/360-System/370. System/360-System/370 can use data on a System/3 data module by treating it like a file. System/3 gives a default filename of SYSTEM/3.DATA. The NAME360 parameter can be used if you would like to code a filename of your own.

NAME360 can contain any of the standard System/3 characters except apostrophes, blanks, and commas. Its length must not exceed 44 characters.

OLDPACK Parameter (VOL)

The OLDPACK parameter (OLDPACK-name) is used to verify that a specific volume is online before initialization is started. If the name of the volume does not match the name you specify, the program halts.

The specified name can be any combination of standard System/3 characters except apostrophes, leading or embedded blanks, and embedded commas. Its length must not exceed 6 characters.

OCL CONSIDERATIONS

The following OCL statements are needed to load the disk initialization program:

```
// LOAD $INIT,code
// RUN
```

The code you supply depends on the location of the simulation area containing the disk initialization program. Possible codes are R1, F1, R2, and F2.

EXAMPLES

Primary Initialization of Two Volumes

Figures 4-67 and 4-68 are examples of OCL statements and control statements needed for the primary initialization of two volumes.

```
// LOAD $INIT,F1
// RUN
```

Explanation:

The disk initialization program is loaded from F1.

Figure 4-67. OCL Load Sequence for Disk Initialization

```
// UIN UNIT-'D1,D2',TYPE-PRIMARY
// VOL PACK-2222
// VOL PACK-PAYROL,ID-120276
// END
```

Explanation:

- The main data areas on two volumes are being initialized (UNIT-'D1,D2' in UIN statement).
- The main data area (D1) is given the name 2222 (PACK-2222 in first VOL statement).
- The main data area (D2) is given the name PAYROL (PACK-PAYROL in second VOL statement). Additional identifying information, 120276, is to be written on drive 2 (ID-120276).

Figure 4-68. Control Statements for Primary Initialization of Two Volumes

MESSAGES FOR DISK INITIALIZATION

Message	Meaning
INITIALIZATION ON ZZZ COMPLETE	This message is printed when initialization of a volume is complete. ZZZ indicates the unit (for example, D31) on which the initialization is complete.
INITIALIZATION ON ZZZ TERMINATED	This message is printed when initialization of a volume must be terminated for one of the following reasons: • Cylinder 0 head 0 is defective. • More than 40 tracks are defective. • Possible disk hardware error exists. After this message is printed, system message 33 occurs. ZZZ indicates the unit (for example, D1) on which the initialization is terminated.
ALTERNATE TRACKS ASSIGNED PRIMARY TRACK XXXX ALTERNATE TRACK YYYY	These two messages are printed when a primary track is defective and an alternate track is assigned to it. XXXX is the number of the defective primary track. YYYY is the number of the assigned alternate track.
ALTERNATE TRACK YYYY DEFECTIVE	This message is printed when an alternate track is defective. YYYY is the number of the defective track.

Messages listed for the alternate track assignment program ($ALT) may be issued when executing TYPE-CLEAR or TYPE-PRIMARY initialization.

Chain Cleaning Program—$KLEAN

PROGRAM DESCRIPTION

The chain cleaning program has the following function:

Exercise the IBM 1403 Printer for the purpose of cleaning the print chain (train).

The print chain is cleaned as each character is printed on a special type-cleaning paper. (See your local IBM representative regarding the availability of this special paper.)

The chain cleaning program can be executed in any partition. However, before this program is executed or before printing occurs if the output is spooled, the 1403 printer ribbon must be removed and the type-cleaning paper installed in place of the current forms. The following PRINTER OCL statement should be used to cause a message for this purpose:

```
// PRINTER DEVICE-1403,FORMSNO-forms number
```

Note: An IMAGE statement must be supplied if the print chain to be cleaned is different than the print chain currently installed on the printer. For more information regarding the IMAGE OCL statement, refer to *IMAGE Statement* in Part 1 of this manual.

OCL CONSIDERATIONS

The following OCL statements are needed to cause a message and load the chain cleaning program:

```
// PRINTER DEVICE-1403,FORMSNO-forms number
// LOAD $KLEAN,code
// RUN
```

The code you supply depends on the location of the simulation area containing the chain cleaning program. Possible codes are R1, F1, R2, F2.

After end of job for $KLEAN, the 1403 print ribbon should be re-installed, the type cleaning paper removed, and the proper forms installed.

File and Volume Label Display Program—$LABEL

PROGRAM DESCRIPTION

The file and volume label display program has the following function:

Display the volume table of contents (VTOC).

This program can be used to:

- Print the entire volume table of contents (VTOC) from an area.
- Print only the VTOC information for certain data files.

In both cases, the program also prints the name of the area.

The printed VTOC information is a record of the contents of the area. There are many reasons why you might need the information. For example:

- Before reinitializing a disk, you want to check its contents to ensure that it contains no libraries, permanent data files, or temporary data files.
- You want to determine the available disk space for libraries or new files.
- You want specific file information, such as the file name, designation (permanent, temporary, scratch), or the space reserved for the file.

The control statements you supply for the program depend on the program use.

STORAGE REQUIREMENTS

The file and volume label display program requires an 8K partition size to display a simulation area VTOC. The program requires a 10K partition size to display a main data area VTOC without sort and a 10K through 18K partition size when sort is used. The amount of storage required when sort is used is a function of the number of entries in the VTOC. The amount of storage required is as follows:

Number of VTOC Entries	Storage Required for Program to Execute with Sort
300	10K
500	12K
700	14K
900	16K
1000	18K

CONTROL STATEMENT SUMMARY

The control statement you must supply depends on the desired results.

Functions	Control Statement ①
Print entire VTOC	// DISPLAY UNIT-code,LABEL-VTOC [,SORT- {NAME / LOCATION}] // END
Print only file information from VTOC	// DISPLAY UNIT-code,LABEL- {filename / filenames} ② // END

① For each use, the program requires the statements in the order they are listed: DISPLAY,END.

② The number of filenames you list for a program run may not exceed 20 (VTOC is considered one filename).

PARAMETER SUMMARY (DISPLAY STATEMENT)

Parameter	Description
UNIT-code	Location of the area containing the VTOC information being printed. Possible codes are R1, F1, R2, F2, and those for the main data areas.
LABEL-VTOC	Print entire contents of VTOC.
LABEL-filename	Print VTOC information for one file.
LABEL-'filename,filename,...'	Print VTOC information for more than one file. ①
SORT-NAME	VTOC information is sorted into alphabetical order by filename.
SORT-LOCATION	VTOC information is sorted into physical location sequence.

① The number of filenames you list for a program run may not exceed 20 (VTOC is considered one filename).

PARAMETER DESCRIPTIONS

UNIT Parameter

The UNIT parameter (UNIT-code) indicates the location of the area containing the VTOC information being printed. Possible codes are R1, F1, R2, F2, and those for the main data areas.

LABEL Parameter

The LABEL parameter indicates the information you want printed: the entire contents of the VTOC or only the information for certain files. The VTOC is an area on disk that contains information about the contents of the area.

SORT Parameter

The SORT parameter can be specified only when LABEL-VTOC is specified. If SORT-NAME is specified, the VTOC information is sorted by filename into alphabetical order. If SORT-LOCATION is specified, the VTOC information is sorted into physical location sequence. This function applies only to main data area VTOCs and requires additional main storage for sorting.

ENTIRE CONTENTS OF VTOC

The parameter LABEL-VTOC means to print the entire contents of the VTOC. The meaning of the information the program prints is given in the following chart. Headings that are listed are the ones printed by the program to identify the information. Figures 4-69 and 4-70 are examples of VTOC printouts.

If the program needs more than one page to list the file information, it prints the headings for the file information at the top of each new page.

```
PACK-F2F2F2  UNIT-F2    DATE-07/10/75    TIME-00.00.57  ID-

DEVICE CAPACITY-400

AVAILABLE SPACE ON PACK
   LOCATION   TRACKS
      008       181

SEQ   FILE   RE-  FILE  FILE  REC KEY  KEY   DATA   FILE  FILE  RECORD  RECORDS   OCL SIZE  NEXT AVAIL    NEXT AVAIL
NUM   NAME  TAIN  DATE  TYPE  LEN LEN  LOC   START   LOC TRACKS  COUNT    AVAIL  PARAMETER    RECORD         KEY

001 DIRF12    T 09/29/71  D 00124                    400  006      297  ******          6T     ******
002 DIR12A    T 07/10/75  D 00124                    394  006      297  ******          6T     ******
003 RTN03B    P 07/10/75  S 00133                    374  020       28     895         20T  374/14/141
004 PDO05B    P 07/10/75  S 00063                    339  035       52    3361         35T  339/12/205
005 SEQ09B    P 07/10/75  S 00512                    329  010       61      59         10T  334/02/001
006 CRT03C    P 07/10/75  S 00133                    309  020       28     895         20T  309/14/141
007 YTD05C    P 07/10/75  S 00063                    274  035       52    3361         35T  274/12/205
008 SEQ09C    P 07/10/75  S 00512                    264  010       61      59         10T  269/02/001
009 SEQ11C    P 07/10/75  S 00128                    258  006       99     189          6T  260/01/129
010 SEQ11C    P 10/30/71  S 00128                    189  007       99     237          7T  191/01/129
011 SEQ11C    P 10/01/71  S 00128                    196  012       98     478         12T  198/01/001
012 ACR08C    T 07/10/75  S 00002                    255  003       54    9162          3T  255/00/109
013 DIR12C    T 07/10/75  D 00124                    249  006      297  ******          6T     ******
014 ACPF07    P 07/10/75  S 00100                    247  002       71      51         71R  248/03/189
015 SEQF09    P 07/10/75  S 00512                    237  010        0     120         10T  237/00/001
016 PRLF08    T 07/10/75  S 00002                    234  003       54    9162          3T  234/00/109
017 PAYF10    T 07/10/75  S 00096                    208  026       38       1         38R  233/08/001
```

Figure 4-69. Simulation Area VTOC Printou

```
PACK-D1D1D1  UNIT-D1    DATE-07/09/75    TIME 00.00.58  ID FIRSTPACK

NO. OF ALTERNATE TRACKS AVAILABLE-40

AVAILABLE SPACE ON PACK

   LOCATION   TRACKS

    002/00    0364

    021/19    0821

    063/02    0756

    101/01    0599

    131/10    0112

SEQ   FILE   RE-  FILE  FILE  REC KEY  KEY   DATA   FILE  FILE  RECORD  RECORDS  OCL SIZE  NEXT AVAIL     NEXT AVAIL
NUM   NAME  TAIN  DATE  TYPE  LEN LEN  LOC   START   LOC TRACKS  COUNT    AVAIL  PARAMETER    RECORD          KEY

001 DIRF12    T 07/09/75  D 00124                  147/06 0002      198  *****          2T     *****
002 DIR12A    T 07/09/75  D 00124                  141/02 0006      594  *****          6T     *****
003 DIR12C    T 07/09/75  D 00124                  137/02 0006      594  *****          6T     *****
004 INDF03    P 07/09/75  I 00133 05 00104 001/02  001/00 0020       28     1635        20T  001/02/15/141  001/00/02/001
005 INDF05    P 07/09/75  I 00063 29 00063 020/17  020/04 0035       52     4239        35T  020/17/13/205  020/04/08/256
006 INDF07    P 07/09/75  I 00100 08 00099 063/01  063/00 0002       65       57        65R  063/01/26/101  063/00/04/025
              VOL SEQ NUM - 01       LOKEY-                                    HIKEY-00000005
                                                                                     0000006F
007 INDF07A   P 07/09/75  I 00100 08 00099 144/07  144/06 0005       71      420         5T  144/07/28/189  144/06/04/097
008 INDF08    T 07/09/75  I 00002 02 00002 101/00  100/18 0003       54     3894         3T  101/00/01/109  100/18/02/073
009 INDF10    T 07/09/75  I 04096 07 04096 144/00  143/19 0004        8        1         8R  144/02/33/001  143/19/01/089
              VOL SEQ NUM - 01       LOKEY-                                    HIKEY-0000008
010 INDF10A   T 07/09/75  I 04096 07 04096 141/09  141/08 0035        0      102        35T  141/09/01/001  141/08/01/001
011 IND03A    P 07/09/75  I 00133 05 00104 146/08  146/06 0020       28     1635        20T  146/08/15/141  146/06/02/001
012 IND03C    P 07/09/75  I 00133 05 00104 140/04  140/02 0020       28     1635        20T  140/04/15/141  140/02/02/001
013 IND05A    P 07/09/75  I 00063 29 00063 145/04  144/11 0035       52     4239        35T  145/04/13/205  144/11/08/256
014 IND05C    P 07/09/75  I 00063 29 00063 139/00  138/07 0035       52     4239        35T  139/00/13/205  138/07/08/256
015 IND08A    T 07/09/75  I 00002 02 00002 144/05  144/03 0003       54     3894         3T  144/05/01/109  144/03/02/073
016 IND08C    T 07/09/75  I 00002 02 00002 137/10  137/08 0003       54     3894         3T  137/10/01/109  137/08/02/073
017 OUT01     T 07/09/75  S 00096                  166/00 0020        1     2559        20T  166/00/01/097
018 OUT02     T 07/09/75  S 00255                  156/00 0020        1      962        20T  156/00/01/256
019 OUT03     T 07/09/75  S 00511                  165/00 0020        1      479        20T  165/00/02/256
020 OUT04     T 07/09/75  S 00096                  155/00 0020        1     2559        20T  155/00/01/097
021 OUT05     T 07/09/75  S 00767                  164/00 0020        1      319        20T  164/00/03/256
022 OUT06     T 07/09/75  S 00020                  154/00 0020        1    12287        20T  154/00/01/021
023 OUT07     T 07/09/75  S 00065                  163/00 0020        1     3779        20T  163/00/01/066
024 OUT08     T 07/09/75  S 00256                  153/00 0020        1      959        20T  153/00/02/001
025 OUT09     T 07/09/75  S 00257                  162/00 0020        1      955        20T  162/00/02/002
026 OUT10     T 07/09/75  S 04096                  152/00 0020        1       59        20T  152/00/17/001
027 OUT11     T 07/09/75  S 00001                  161/00 0020        1   245759        20T  161/00/01/002
028 OUT12     T 07/09/75  S 00100                  151/00 0020        1     2456        20T  151/00/01/101
029 OUT13     T 07/09/75  S 00010                  160/00 0020        1    24575        20T  160/00/01/011
030 OUT14     T 07/09/75  S 00127                  150/00 0020        1     1934        20T  150/00/01/128
031 OUT15     T 07/09/75  S 00128                  159/00 0020        1     1919        20T  159/00/01/129
032 OUT16     T 07/09/75  S 00128                  149/00 0020        1     1919        20T  149/00/01/129
033 OUT17     T 07/09/75  S 01024                  158/00 0020        1      239        20T  158/00/05/001
034 OUT18     T 07/09/75  S 00032                  148/00 0020        1     7679        20T  148/00/01/033
035 OUT19     T 07/09/75  S 00064                  157/00 0020        1     3839        20T  157/00/01/065
036 SEQF09    P 07/09/75  S 00512                  131/00 0010        0      240        10T  131/00/01/001
037 SEQF11    P 07/09/75  S 00128                  147/08 0012       97     1055        12T  147/09/01/129
038 SEQ09A    P 07/09/75  S 00512                  143/09 0010       61      179        10T  143/11/27/001
039 SEQ09C    P 07/09/75  S 00512                  137/17 0010       61      179        10T  137/19/27/001
040 SEQ11A    P 07/09/75  S 00128                  143/03 0006        0      576         6T  143/03/01/001
041 SEQ11C    P 07/09/75  S 00128                  137/11 0006       99      477         6T  137/12/02/129
```

Figure 4-70. Main Data Area VTOC Printout

MEANING OF VTOC INFORMATION

Heading	Meaning
PACK-name	Name of the area.
UNIT-code	Location of the area containing the VTOC information.
DATE-xx/xx/xx	Partition date.
TIME-xx.xx.xx	Time of day.
ID-characters	Additional area identification (if any).
NUMBER OF ALTERNATE TRACKS AVAILABLE-number	Number of alternate tracks available for assignment. Main data area only.
TRACKS WITH ALTERNATE ASSIGNED	Address of primary tracks that have been assigned an alternate. Main data area only.
DEFECTIVE ALTERNATE TRACKS	Address of the alternate tracks that are defective. Main data area only.
DEVICE CAPACITY-number	Disk capacity (number of tracks). Simulation area only.
LIBRARY EXTENT	Boundary of libraries on the simulation area. (If the simulation area contains no libraries, these headings are not printed.)
START	Track on which library begins. If the simulation area contains both source and object library, START refers to beginning of source library and END refers to end of object library.
END	Track on which library ends.
EXTENDED END	Object library only (simulation area only). Track on which extension to library ends. When object library is full, temporary entries can be placed in space following end of library, provided that space is available.
AVAILABLE SPACE ON PACK	Available disk space.
LOCATION	First track in available space (simulation area). First cylinder/track in available space (main data area).
TRACKS	Number of tracks available.
SEQ NUM	Line number.
FILE NAME	Name that identifies file in VTOC.
RETAIN	File designation: P = Permanent T = Temporary S = Scratch (simulation area only)

Heading	Meaning
FILE DATE	Date given the file when file was placed on disk.
FILE TYPE	File type: I = indexed S = sequential D = direct * = file used by spooling
REC LEN	Number of characters in each record in file.
KEY LEN	Number of characters in each record key (indexed files only).
KEY LOC	Position in record occupied by last character of record key (indexed files only).
DATA START	Disk space reserved for indexed files only. START is the first main data area cylinder/track of the area. This refers to the data portion of the file.
FILE LOC	First track used by the file. For simulation area files, refers to a track number. For main data area files, refers to a cylinder/track number.
FILE TRACKS	Number of tracks allocated to the file.
RECORD COUNT	Total number of records currently in the file.
RECORDS AVAIL	Number of records that can be added to the file. (1)
OCL SIZE PARAMETER	Parameter used on OCL statement when file was created. T = tracks R = records
NEXT AVAIL RECORD	Beginning location of next available record in file. For simulation area, location is track, sector, and position within sector. For main data area, location is cylinder, track, fixed record, and position within record. Example: 099/18/006 = track 99, sector 18, positions 6. (1) 050/02/12/006 = cylinder 50, track 2, fixed record 12, position 6. (1)
NEXT AVAIL KEY	Indexed files only. Beginning location of next available record key in index portion of file. For main data area, location is cylinder, track, fixed record, and position within record. Main data area only. Example: 052/03/10/006 = cylinder 52, track 3, fixed record 10, position 6. (1)
VOL SEQ NUM	VOL SEQ NUM applies to multivolume files only. It indicates the order of the volume as it relates to the other volumes containing the remaining portion of the file. Main data area only.

Heading	Meaning
LOKEY	The high key from the previous volume. This field will be blank for the first volume of a multivolume file. Main data area only. (2)
HIKEY	The highest key that can be put on any specific volume of a multivolume indexed file. Main data area only. (2)

(1) If the field contains *****, there is insufficient space in the file for additional records or index entries.

(2) A packed key is printed on two print lines. For example, a packed key consisting of 0000125F is printed as 0015 / 002E.

FILE INFORMATION ONLY

The parameter LABEL-filename or LABEL-'filenames' means to print certain file information from the VTOC. For one file, use LABEL-filename; for two files, use LABEL-'filename,filename'; and so on. (Use the names that identify the files in the VTOC.) You can list 20 file-names for a program run. The statement length, however, is restricted to 96 characters, and continuation statements are not supported.

The program prints the file information for each of the files you list. This information is described under *Meaning of VTOC Information.*

If the program needs more than one page to list the file information, it prints headings for the file information at the top of each new page.

The ALLOCATED SPACE NOT EQUAL TO SPACE USED message is printed for one or more of the following reasons:

- The VTOC is not updated, even though space has been allocated for a file, because of an abnormal termination of an executing program.
- A consecutive or direct file is built over an indexed file (load to old). The new file overlays the data portion of the indexed file leaving the index portion unaltered.
- A checkpointed program has not been completed.

Unused space can be made available by use of the FORMAT statement (file delete program, $DELET), except when a consecutive or direct file is built over an indexed file.

OCL CONSIDERATIONS

The following OCL statements are used to load the file and volume label display program.

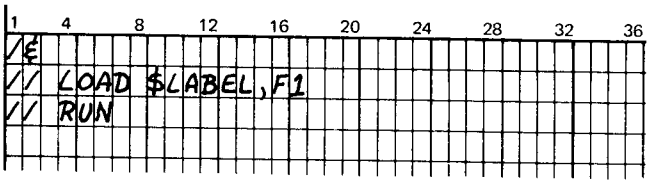
```
// LOAD $LABEL,code
// RUN
```

The code you supply depends on the location of the simulation area containing the system service program. Possible codes are R1, F1, R2, and F2.

EXAMPLE

Printing VTOC Information for Two Files

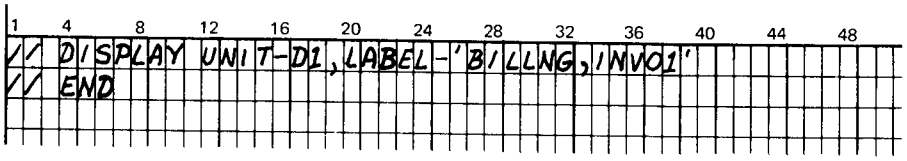
Figures 4-71 and 4-72 are examples of the OCL statements and control statements needed to print VTOC information for two files.



Explanation:

- The file and volume label display program is loaded from F1.

Figure 4-71. OCL Load Sequence for File and Volume Label Display



Explanation:

- The files for which information is printed are named BILLNG and INVO1 (LABEL-'BILLNG, INVO1' in DISPLAY statement). They are located on main data area D1 (UNIT-D1).

Figure 4-72. Control Statements for Printing VTOC Information for Two Files

Library Maintenance Program—$MAINT

PROGRAM DESCRIPTIONS

The library maintenance program has the following functions:

- Allocate library space for user and system libraries.
- Copy entries to, and display the contents of, libraries. Create a file from library entries.
- Delete library entries.
- Modify source library entries.
- Rename library entries.

The control statements you must supply depend on the function you are using.

For further information about library format, organization, and location refer to *Library Facilities* in Part 2 of this manual.

For information concerning multiprogramming considerations, see *Library Maintenance Program* under *Multiprogramming Considerations and Restrictions* in Part 2 of this manual.

Use of Disk Space

The library maintenance program acquires library space on disk during each of the following functions:

- Create a library.
- Increase the size of a library.
- Dynamically extend an object library to copy temporary entries to the library.

If there is not a library on the simulation area, the space used to create one is the first contiguous space large enough to contain the library. If there is a library, the contiguous space directly following it will be used.

The library maintenance program uses temporary work space on disk for each of the following functions:

- Reorganize a library.
- Sort a directory before it is printed.
- Modify a source library entry.

The work space must be on a simulation area except when the program is reorganizing a library, in which case it can also be on a main data area.

To sort a library directory before printing, the library maintenance program requires a work space, on the simulation area specified by the FROM parameter, equal to the size of the directory. Thus, if your object library directory occupies, for example, six tracks, a work space of six unused tracks is required to sort that directory. When the function is complete, the work space is released and may be used by another library maintenance function, or by a program allocating files in another partition.

ORGANIZATION OF THIS SECTION

The five functions of the library maintenance program are described separately. Every description contains the following:

- List of specific uses.
- Control statement summary indicating the form of control statement needed for each use.
- Parameter descriptions explaining in detail the contents and meanings of the parameters.
- Function descriptions explaining the details of each function.

Following the function descriptions are OCL considerations and examples.

$MAINT—Allocate Function

USES

- Create (reserve space for) libraries, system history area, scheduler work area, and checkpoint/restart area.
- Change the size of libraries and system history area.
- Delete libraries.
- Reorganize libraries.

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

All volumes referenced by the control statements must remain online during the execution of $MAINT.

// ALLOCATE TO-code,SOURCE-{number / R},OBJECT-{number / R},SYSTEM-{NO / YES}[,HISTORY-number]
[,DIRSIZE-number] [,WORK-code] [,PACKO-name]

	Use (1)	Parameter Needed (2)
Source Library	Create	TO-code,SOURCE-number,WORK-code (3)
	Change size	TO-code,SOURCE-number,WORK-code
	Delete	TO-code,SOURCE-0
	Reorganize	TO-code,SOURCE-R,WORK-code
Object Library	Create	TO-code,OBJECT-number,SYSTEM-{NO / YES}
	Change size	TO-code,OBJECT-number,WORK-code (4)
	Delete	TO-code,OBJECT-0
	Reorganize	TO-code,OBJECT-R,WORK-code (4)

(1) You can indicate a source library use, any object library use, or uses involving both libraries (for example, deleting the source library and changing the size of the object library).

(2) If you are indicating uses for both libraries, use only one TO parameter. (The libraries must be on the same simulation area.) Also, use only one WORK parameter if both uses require a WORK parameter.

(3) The WORK parameter is needed only if the simulation area contains an object library that you are not deleting.

(4) The WORK parameter is not required if this is a compress in place.

CONSIDERATIONS AND RESTRICTIONS

- The allocate function cannot reference the libraries on the simulation area from which the library maintenance program or the system was loaded. For example, if the system was loaded (IPL) from F1 and the library maintenance program was loaded from R1, the source or object libraries on F1 and R1 cannot be referenced on an ALLOCATE statement.

- When a library is reorganized, its size is changed, or it is moved, all temporary entries in that library are deleted. This deletion applies to both the source and object libraries.

- When you are creating or changing the size of the source library on a simulation area that contains an object library, the object library is moved and reorganized and all temporary entries are deleted.

- The SOURCE or OBJECT parameter must be specified in the ALLOCATE statement. If the SYSTEM, DIRSIZE, or HISTORY parameter is specified, the OBJECT parameter must also be specified.

- If nested procedures are used, information contained in the scheduler work area can become invalid when the source library is reorganized or the size of the source library is changed (reallocated). Therefore, if you use a procedure to reorganize or reallocate libraries, do not call any further procedures contained within that nested procedure from the source library that is being reallocated or reorganized.

- Information can be lost from the system history area (SHA): when an object library is reorganized (unless *compress in place* is used), when the source library is allocated, or when the size of the source library is changed.

PARAMETER SUMMARY

Parameter	Description
TO-code	The simulation area that contains or will contain the library. Possible codes are R1, F1, R2, F2.
SOURCE-number (no source library exists)	Create a source library. Number indicates the number of tracks you want to assign.
SOURCE-number (source library already exists)	Delete or change the size of the source library. Use depends on number: *Number* — *Use* 0 — Delete Any number but zero — Change size
SOURCE-R	Reorganize the source library.
OBJECT-number (no object library exists)	Create an object library. Number indicates the number of tracks you want to assign.
OBJECT-number (object library already exists)	Delete or change the size of the object library. Use depends on number: *Number* — *Use* 0 — Delete Any number but zero — Change size
OBJECT-R	Reorganize the object library.
DIRSIZE-number	Number of tracks you want for the directory when creating, reallocating, or reorganizing the object library.
SYSTEM-NO	Do not create a scheduler work area. This will be a program pack.
SYSTEM-YES	Create a scheduler work area and a system history area. This will be a system pack.
HISTORY-number	Number of tracks you want for a system history area.
WORK-code	The area containing space the program can use as a work area. Possible codes are R1, F1, R2, F2 and those for the main data areas.
PACKO-name	Name of the simulation area specified by the TO parameter.

PARAMETER DESCRIPTIONS

TO Parameter

The TO parameter (TO-code) indicates the location of the simulation area that contains, or will contain, the library. If the program use involves both libraries, the libraries must be on the same simulation area. The TO parameter cannot be the same unit from which the library maintenance program or system is loaded. Possible codes are R1, F1, R2, F2.

SOURCE and OBJECT Parameters

These parameters identify library uses:

Parameter	Use
SOURCE-number OBJECT-number (number is not zero)	If the simulation area contains no library, this parameter means create a library. Number is the number of tracks you want to assign to the library. If the simulation area contains a library, this parameter means change the library size. Number is the number of tracks you want to assign to the library.
SOURCE-0 OBJECT-0	Delete the library.
SOURCE-R OBJECT-R	Reorganize the library.

DIRSIZE Parameter

The DIRSIZE parameter allows the user to specify the size of the object library directory. The number of tracks specified (1–9) overrides the SYSTEM parameter in determining directory size. Each track can contain 288 directory entries. One entry is needed for the directory, so the formula for the number of entries in a directory is (t x 288) – 1, where t is the number of tracks. If the DIRSIZE parameter is omitted, the SYSTEM parameter determines the directory size.

SYSTEM Parameter

The SYSTEM parameter applies when you are creating, changing the size of, or reorganizing object libraries. It indicates to the program whether you intend to include system programs in the library to create a system pack from which an IPL may be performed. If system programs are to be included, a scheduler work area must be assigned. See *Copy Function (Library-to-Library)* in this manual for information about creating a system pack.

Space for the scheduler work area is assigned immediately preceding the object library. If the simulation area contains a source library, the scheduler work area is between the source and object libraries. For information about the size of the scheduler work area, see *Using the Allocate Function.*

The system history area is allocated a space preceding the scheduler work area.

The following charts show the results of coding the SYSTEM parameter for different allocate uses.

Creating an Object Library

Parameter	Scheduler Work Area	Directory Size[1]	System History Area
SYSTEM-YES	Created	3 tracks	Created
SYSTEM-NO	Not created	1 track	Not created
Not coded	Not created	1 track	Not created

[1] The directory size is overridden if the DIRSIZE parameter is coded.

Changing the Size of or Reorganizing an Object Library on a Simulation Area That Contains a Scheduler Work Area and a System History Area

Parameter	Scheduler Work Area	Directory Size[1]	System History Area
SYSTEM-YES	Retained	Not changed	Retained (existing entries may be lost)
SYSTEM-NO	Removed	Not changed	Removed
Not coded	Retained	Not changed	Retained (existing entries may be lost)

[1] The directory size is overridden if the DIRSIZE parameter is coded.

Changing the Size of or Reorganizing an Object Library on a Simulation Area That Does Not Contain a Scheduler Work Area and a System History Area

Parameter	Scheduler Work Area	Directory Size[1]	System History Area
SYSTEM-YES	Created	Not changed	Created
SYSTEM-NO	Not created	Not changed	Not created
Not coded	Not created	Not changed	Not created

[1] The directory size is overridden if the DIRSIZE parameter is coded.

HISTORY Parameter

Every system pack must have a system history area (SHA), which is used to store OCL, OCC, system messages, message responses, control statements, and other information (see *$HIST*). The HISTORY parameter in the ALLOCATE statement allows you to specify the size of the SHA on the system pack. The minimum size is 2 tracks; the maximum is the number of tracks in the available area. If HISTORY is not specified when you are allocating space on a simulation area, the size defaults to 2 tracks. The size of the SHA remains unchanged, unless it is altered by a HISTORY parameter. See *System History Area* under *System Facilities* in Part 2 of this manual for additional information in determining the number of tracks to be specified for the HISTORY parameter.

WORK Parameter

The WORK parameter (WORK-code) indicates the location of the area that contains a work area. Library entries are temporarily stored in the work area while the program moves and reorganizes libraries. Possible codes are R1, F1, R2, F2, and those for the main data areas.

Size of the Work Area

The work area must be large enough to hold the directory and the permanent entries of the source library, object library, or both libraries depending on the program use. If you are combining uses, such as changing the sizes of both libraries, the work area must be large enough to hold the contents of both libraries.

Use	Contents of Work Area
Create a source library (an object library exists).	Object library
Change source library size (an object library exists).	Source library and object library
Change source library size (an object library does not exist).	Source library
Reorganize source library.	Source library
Change object library size.	Object library, if not compress in place (see *Compress in Place.*)
Reorganize object library.	Object library, if not compress in place (see *Compress in Place.*)

If the WORK parameter is supplied and is not required, it will be ignored and the message WORK PARAMETER IGNORED will be logged.

Location of Work Space

The program uses the last available space large enough to hold the library or libraries.

Location of Area Containing the Work Space

The work space can be on any area on any volume. However, it cannot be the same simulation area as the one you specified in the TO parameter. The only requirement is that the area have space available large enough for the work area. The program works faster if the simulation area containing the libraries is on a different drive than the area containing the work space.

PACKO Parameter

The optional PACKO parameter (PACKO-name) verifies that the correct library is being accessed before the program performs the allocate function. If supplied, the program compares the name given in the PACKO parameter with the name in the volume label on the simulation area specified by the TO parameter to ensure that they match. If not supplied, no verification is done.

The name can be any combination of standard System/3 characters except apostrophes, leading or embedded blanks, and embedded commas. Its length must not exceed 6 characters. See Appendix A for a list of standard System/3 characters.

USING THE ALLOCATE FUNCTION

Creating a Source Library (SOURCE-number)

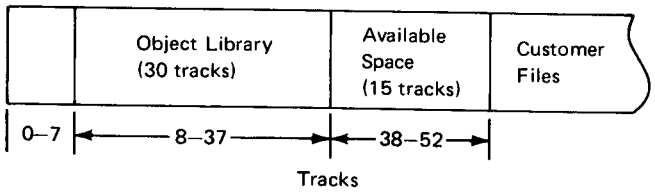
Source Library Size

- Minimum: One track.
- Maximum: Number of tracks in the available area.
- Regardless of the number of tracks you specify, the first two sectors of the first track are assigned to the library directory. Additional sectors are used as needed for the directory.

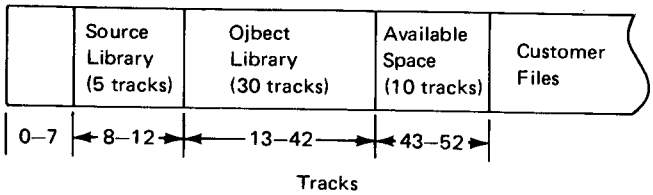
Placement of Source Library (An Object Library Exists):

- The source library must precede the object library. A disk area large enough for the source library must follow the object library because the program moves the object library to make room for the source library. To do this, it needs a work area (see *WORK Parameter*). The object library is reorganized, and all temporary entries are deleted.
- If you allocate a source library after deleting it, the program automatically moves the object library to make room for the source library. The starting location of the source library is the previous starting location of the object library.

Disk Space Before Creating Source Library



Disk Space After Creating Source Library



Placement of the Source Library (An Object Library does not Exist): The program assigns the source library to the first available disk area large enough for the library.

If you allocate a source library after deleting it, the source library is assigned the same way.

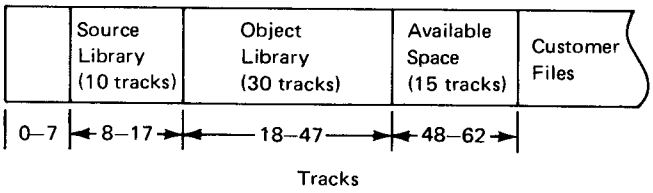
Changing the Size of (Reallocating) a Source Library (SOURCE-number)

Any time the program changes the source library size, it reorganizes both the source and object libraries and deletes all temporary entries. (See *Reorganizing a Source Library.*) To do this, it needs a work area. (See *WORK Parameter.*)

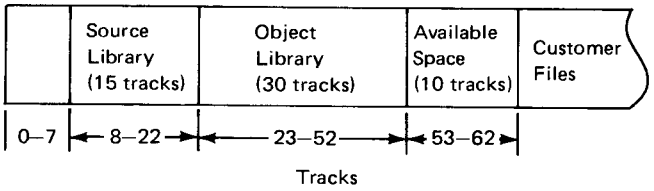
Making the Source Library Larger

- If the simulation area contains an object library, space must be available immediately following the object library. The program moves the object library to make tracks available at the end of the source library.
- If the simulation area does not contain an object library, space must be available immediately following the source library.

Disk Space Before Tracks Are Added to Source Library



Disk Space After 5 Tracks Are Added to Source Library

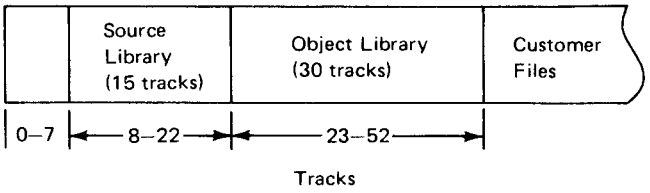


Making the Source Library Smaller

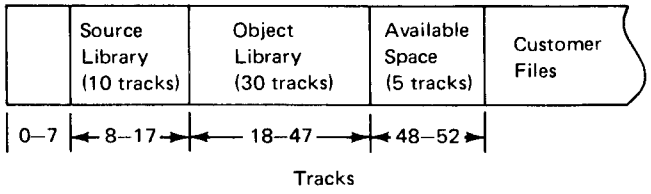
- If the simulation area contains an object library, the program moves the end location of the source library to make the library smaller. The object library is moved, and space becomes available following the object library.

- If the simulation area does not contain an object library, the program moves the end location of the source library to make the source library smaller.

Disk Space Before Source Library Size Was Decreased



Disk Space After 5 Tracks Were Taken From Source Library



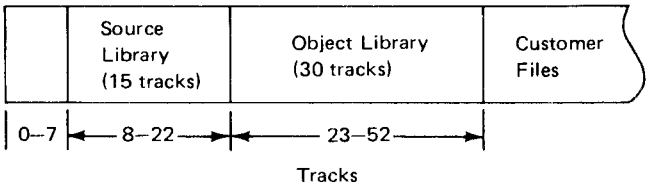
Deleting a Source Library (SOURCE-0)

The library maintenance program makes the disk area occupied by the source library available for other use (disk files) by moving the starting location of the library to the beginning of the object library.

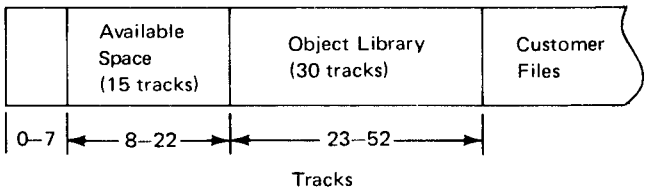
Then, if you try to recreate the source library, $MAINT checks for sufficient library space for the source and the object libraries from the beginning of the object library.

If you want to retain the deleted space for library usage, the OBJECT-R parameter should be used in addition to the SOURCE-0 parameter on the same ALLOCATE statement. Then, the starting location of the object library is moved to the starting location formerly occupied by the deleted source library.

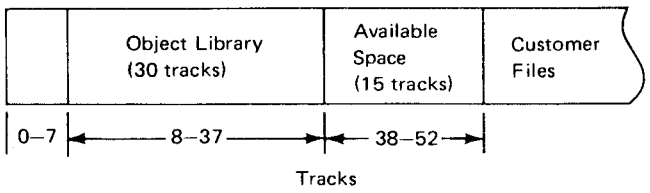
Disk Space Before Source Library Was Deleted



Disk Space After Source Library Was Deleted (SOURCE-0 only)



Disk Space After Source Library Was Deleted With Object Library Reorganized (SOURCE-0 and OBJECT-R)



Reorganizing a Source Library (SOURCE-R)

Reason for Reorganizing the Library

Areas from which source library entries are deleted are completely reused for new entries. If an entry exceeds the space in such an area, the program puts as much of the entry as will fit in the area and continues the entry in the next available area. In this way, the program efficiently uses library space. This technique however, may decrease the speed at which those entries can be read from the library. Therefore, if you frequently add or delete source library entries, you should reorganize your source library periodically.

Reorganizing the Library

The program relocates entries so that no entry is started in one area and continued in another. All temporary entries are deleted. The program needs a work area. (See *WORK Parameter.*)

Creating an Object Library (OBJECT-number)

Object Library Size

- Minimum: Three tracks including the directory tracks.
- Maximum: Number of tracks in available area.
- Library Directory: The first 3 tracks in the library are reserved for the library directory if the library is to contain system programs; otherwise, only the first track is used. If the DIRSIZE parameter is entered, the directory size specified is used.
- Scheduler Work Area. The scheduler is a component of the System/3 SCP that reads and processes OCL statements. It uses a work area on disk (SWA) to temporarily save OCL file label information during the processing of a program. The area is allocated when SYSTEM-YES is specified. The work space is not included in the number you specify in the OBJECT parameter; the space is calculated and assigned by the library maintenance program. The minimum space required for the scheduler work area is 18 tracks. If you generated your system to support checkpoint/restart, the size of the scheduler work area is 33 tracks (including a checkpoint/restart area).

Scheduler Work Area without Checkpoint/ Restart	Scheduler Work Area with Checkpoint/ Restart
18 tracks	33 tracks

The scheduler work area contains F1 and F7 label information, an initiator table, control statement area, and miscellaneous work areas. There is one scheduler work area for each partition. (See *Scheduler Work Area* under *File Facilities* in Part 2 of this manual.)

Placement of Object Library (A Source Library Exists): Space for the object library must be available immediately following the source library.

Placement of Object Library (A Source Library Does Not Exist): The program assigns the object library to the first available space that is large enough.

Changing the Size of (Reallocating) an Object Library (OBJECT-number)

Making the Library Larger

The number of tracks you want to add must be available immediately following the object library. The program assigns the additional tracks to the library. (The starting location of the library remains unchanged.)

Making the Library Smaller

The program moves the end location of the object library to decrease the library size. Tracks, therefore, become available following the library.

Reorganizing the Library

Any time the program changes the library size it also reorganizes the library and deletes all temporary entries (see *Reorganizing an Object Library*). A work area is needed if other functions are being performed with the reorganization (see *WORK Parameter*). If not, a work area is not used (see *Compress in Place*).

Deleting an Object Library (OBJECT-0)

The program makes the disk space occupied by the object library (and the scheduler work area and the system history area if this was a system pack) available for other use.

Reorganizing an Object Library (OBJECT-R)

Gaps can occur between object library entries when you add and delete entries. By reorganizing the library, these gaps are removed. When the library is reorganized, all temporary entries are deleted. A work area is needed if other functions are being performed with the reorganization (see *WORK Parameter*). If not, a work area is not used (see *Compress in Place*).

Compress in Place (OBJECT-{R | number})

If an object library is to be reorganized, or the size is to be changed and this is the only function to be performed, the object library will be compressed in place. This means that the library will be reorganized with all gaps removed and all temporary entries deleted without the use of a work area. If supplied, the WORK parameter will be ignored and the message WORK PARAMETER IGNORED will be logged.

If, however, a source library function is to be performed or if the directory size (DIRSIZE parameter) or the pack type (SYSTEM parameter) or the system history area size (HISTORY parameter) is to be changed in conjunction with an object library function, a work area will be used (see *WORK Parameter*).

$MAINT—Copy Function

USES

Use	Description
Reader-to-Library	• Add or replace a library entry. The reader is the system input device.
File-to-Library	• Add or replace one or more library entries. A disk file is the input.
Library-to-File	• Copy one or more library entries to a disk file.
Library-to-Library	• Copy one library entry (or those entries with the same name from all libraries). • Copy library entries that have names beginning with certain characters. • Copy all library entries. • Copy minimum system.
Library-to-Printer	• Print one library entry (or those entries with the same name from all libraries). • Print library entries that have names beginning with certain characters. • Print all library entries of a certain type. • Print directory entries for library entries of a certain type. • Print entries from all directories including system directory. • Print system directory only.

Use	Description
Library-to-Card or Diskette	• Punch one library entry (or those entries with the same name from all libraries). • Punch library entries that have names beginning with certain characters. • Punch all library entries of a certain type.
Library-to-Printer and-Card or Diskette	• Print and punch one library entry (or those entries with the same name from all libraries). • Print and punch library entries that have names beginning with certain characters. • Print and punch all temporary or permanent library entries of a certain type.

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

All volumes referenced by the control statements must remain online during the execution of $MAINT.

Reader-to-Library

Add or Replace a Library Entry

// *COPY FROM-READER*,LIBRARY-{S/P/O/R},NAME-name,

TO-code [,RETAIN-{T/P/R}] [,PACKO-name]

-
-
library entry
-
-

// CEND ◢ Must always follow the source or object entry being placed into the source or object library.

/*, /& or /. statements cannot be present in the entries being copied into the libraries.

File-to-Library

Add or Replace One or More Library Entries

// COPY FROM-DISK,FILE-filename,TO-code

[,RECL-{80/96}] [,RETAIN-{T/P/R}]

[,PACKO-name]

Example of Data in Disk File

① // COPY FROM-READER,LIBRARY-O,RETAIN-P,
NAME-DECK01
-
-
load module
-
-
// CEND

① // COPY LIBRARY-S,NAME-DECK02
-
-
source module
-
-
// CEND

② // END

Library-to-File

Copy One or More Library Entries to a File

// COPY FROM-code,TO-DISK,FILE-filename

[,RECL-{80/96}] [,PACKIN-name]

Control Statements Following // COPY

// ENTRY LIBRARY-{S/P/O/R/ALL},NAME-{name/characters.ALL/ALL} ③

// NEND ◢ Required to terminate the copy to file.

---

① Only the LIBRARY and NAME parameters are required. Other parameters are ignored.

② The // END statement read from the file is optional. It causes the next statement to be read from the system input device or procedure. A // END statement must still be read from the system input device or procedure to indicate the end of the library maintenance control statements.

③ Any number of // ENTRY statements may precede the // NEND statement.

Library-to-Library

Copy One Library Entry (or Entries with the Same Name from All Libraries)

```
// COPY FROM-code,LIBRARY-{S|P|O|R|ALL},NAME-name,
      TO-code [,RETAIN-{T|P|R}] [,NEWNAME-name]
      [,PACKIN-name] [,PACKO-name] (1)
```

Copy Library Entries that Have Names Beginning with Certain Characters

```
// COPY FROM-code,LIBRARY-{S|P|O|R|ALL}
      ,NAME-characters.ALL,TO-code
      [,RETAIN-{T|P|R}] [,NEWNAME-characters]
      [,PACKIN-name] [,PACKO-name]
```

Copy All Library Entries

```
// COPY FROM-code,LIBRARY-{S|P|O|R|ALL},NAME-ALL,
      TO-code [,RETAIN-{T|P|R}] [,PACKIN-name]
      [,PACKO-name] (1)
```

Copy Minimum System

```
// COPY FROM-code,LIBRARY-O,NAME-SYSTEM,
      TO-code [,PACKIN-name] [,PACKO-name] (1)
```

---

(1) NEWNAME parameter is needed in any of the following cases:

- If you want the copy to have a different name than the original entry.
- If you want to replace an entry on the simulation area specified by the TO parameter with an entry from the simulation area specified by the FROM parameter, but the entries have different names.
- If you want the names of the copies to begin with different characters than the names of the original entries, the same number of characters must be in the NEWNAME parameter as in the NAME parameter.
- If the FROM and TO codes are the same.

Note: NEWNAME cannot be DIR, ALL, or SYSTEM.

(1) FROM and TO parameters cannot have the same code.

Library-to-Printer-and/or-Card

Print and/or Punch One Library Entry (or Entries with the Same Name from All Libraries)

// COPY FROM-code,LIBRARY-{S / P / O / R / ALL},NAME-name,

TO-{PRINT / PUNCH / PRTPCH}[,PACKIN-name]

Print and/or Punch Temporary and Permanent Library Entries that Have Names Beginning with Certain Characters

// COPY FROM-code,LIBRARY-{S / P / O / R / ALL}

,NAME-characters.ALL,TO-{PRINT / PUNCH / PRTPCH}

[,PACKIN-name]

Print and/or Punch All Temporary and Permanent Library Entries of a Certain Type

// COPY FROM-code,LIBRARY-{S / P / O / R},NAME-ALL,

TO-{PRINT / PUNCH / PRTPCH}[,PACKIN-name]

Print Directory Entries for Library Entries of a Certain Type

// COPY FROM-code,LIBRARY-{S / P / O / R},NAME-DIR,

TO-PRINT[,PACKIN-name] [,CDATE-{YES / NO}] ①

Print Entries from All Directories Including System Directory

// COPY FROM-code,LIBRARY-ALL,NAME-DIR,

TO-PRINT[,PACKIN-name] [,CDATE-{YES / NO}] ①

Print System Directory Entries Only

// COPY FROM-code,LIBRARY-SYSTEM,NAME-DIR,
TO-PRINT[,PACKIN-name]

Print Directory Entries, Omitting Selected Entries

// COPY FROM-code,LIBRARY-{S / P / O / R / ALL},NAME-DIR,

TO-PRINT OMIT-{name / characters.ALL}

[,PACKIN-name]

CONSIDERATIONS AND RESTRICTIONS

- When the library-to-file function is used, the FILE parameter on each COPY control statement must refer to a unique filename. Otherwise, previous entries in the file will be overlaid.

① The CDATE parameter is valid only if LIBRARY-O or LIBRARY-ALL is specified.

PARAMETER SUMMARY

Parameter	Description
FROM-READER	Entry to be placed in library is to be read from the system input device.
FROM-code	The simulation area containing library entries to be copied, printed, or punched. Possible codes are R1, F1, R2, F2.
FROM-DISK	The entry or entries to be placed into a library or libraries reside in a disk file. The disk file must be described by an OCL FILE statement
FILE-filename	For a file-to-library or library-to-file copy, this parameter is needed to identify the file. The filename must match the filename on the OCL FILE statement.
RECL-{80 / 96}	For a file-to-library or library-to-file copy, this parameter gives the size of the disk records. Only 80- or 96-column card image records are allowed. If this parameter is omitted, 96 is assumed.
LIBRARY-{S / P / O / R}	Type of library entries involved in copy use. Possible codes are: **Code** — **Meaning** S — Source statements (source library) P — OCL procedures (source library) O — Object programs (object library) R — Routines (object library)
LIBRARY-ALL	All types of entries (S, P, O, and R) from both libraries are involved in the copy use.
LIBRARY-SYSTEM	Only system directory entries will be printed.
NAME-{name / characters.ALL / ALL}	Specific library entries on the FROM simulation area, of the type indicated in LIBRARY parameter, involved in copy use. Possible information is: **Information** — **Meaning** name — Name of the library entry involved. characters.ALL — Only those entries beginning with the indicated characters. For example, $MA.ALL means the library maintenance program ($MAINT). ALL — All entries (the type indicated in LIBRARY parameter).

Parameter	Description
NAME-SYSTEM	System programs that make up the minimum system and IPL information contained on cylinder 0 are copied. The minimum system is made up of system programs necessary to load and run programs. System programs necessary to generate and maintain the system, such as system service programs, are not included in the minimum system.
NAME-DIR	Directory entries for all library entries of the type indicated in the LIBRARY parameter are involved in the copy use. If the LIBRARY parameter is LIBRARY-ALL, system directory entries are also printed.
RETAIN-{T/P/R}	*Adding Entry to Library.* RETAIN gives designation of the TO entry: **Code** — **Meaning** T — Temporary P or R — Permanent *Replacing Existing Library Entry.* RETAIN gives designation of the TO entry and tells the program whether to issue a message before replacing entry: **Code** — **Meaning** T — Temporary designation. Issue message before replacing entry. P — Permanent designation. Issue message before replacing entry. R — Permanent designation. Do not issue message before replacing entry. *Printing or Punching Entries.* The RETAIN parameter is ignored.
TO-code	The simulation area that is to contain the copies of the entries. Possible codes are R1, F1, R2, F2.
TO-PRINT	Entries are printed.
TO-PUNCH	Entries are punched or written to diskette (the diskette device must be specified as the PUNCH device).
TO-PRTPCH	Entries are printed and punched, or printed and written to diskette (the diskette device must be specified as the PUNCH device).
TO-DISK	The entry or entries are to be copied to a disk file. The disk file must be described by an OCL FILE statement.

Parameter	Description
NEWNAME-name	Name you want used on the simulation area specified by the TO parameter to identify the entries being put on that simulation area. If you omit this parameter, the program uses the NAME parameter in naming the entries.
NEWNAME-characters	Beginning characters you want to use in names identifying entries being put on TO simulation area. You must use the same number of characters as in the NAME parameter (NAME-characters.ALL). If you omit this parameter, the program uses the NAME parameter in naming the entries.
OMIT-name	When printing directory entries, omit the entry specified by *name.*
OMIT-characters.ALL	When printing directory entries, omit all entries with these beginning characters.
PACKO-name	Name of the simulation area specified by the TO parameter.
PACKIN-name	Name of the simulation area specified by the FROM parameter.
CDATE-YES / NO (NO underlined)	The optional CDATE-YES parameter specifies that the compilation date and time for the COBOL, FORTRAN, and RPG programs are to be printed; this is the same date and time that appeared in the heading during compilation (time is shown only if timer support was generated on the system doing the compilation). When CDATE-YES is specified, execution time may be increased because $MAINT must obtain the date and time from the object program. The date and time are not printed if CDATE-NO is specified or assumed. (The CDATE parameter, YES or NO, is valid only when LIBRARY-O or LIBRARY-ALL is specified.)

LIBRARY DIRECTORIES

Source and Object Library Directories

- The source and object libraries have separate library directories. Every library entry has a corresponding entry in its library directory. The directory entry contains such information as the name and location of the library entry. (See Figures 4-73 and 4-74.)

- The library maintenance program makes an entry in the directory after it has put the entry in the library.

System Directory

- Each simulation area that contains libraries contains a system directory. The system directory contains information about the sizes of and available space in libraries and their directories. (See Figure 4-75.)

- The library maintenance program creates and maintains the system directory.

NAMING LIBRARY ENTRIES

Characters to Use

Use any combination of System/3 characters except blanks, commas, quotes, and periods (Appendix A lists the characters). The names of most IBM programs begin with a dollar sign ($). Therefore, to avoid possible duplication, do not use a dollar sign as the first character in the names you use for your entries. The first character must be alphabetic.

Length of Name

The name can be from 1 to 6 characters long.

Restricted Names

Do not use the names ALL, DIR, and SYSTEM. They have special meanings in the NAME parameter.

Entries with the Same Name

For each of the two physical libraries, source and object, there are two types of entries. The source library has type P and type S entries. The object library has type O and type R entries. Entries of the same type cannot have the same name, but entries of different types may. For example, two procedures in a source library cannot have the same name, but a procedure and a set of source statements can.

RETAIN TYPES

Temporary Entries

- Temporary entries are entries you do not intend to keep in your libraries. They are normally used only once or a few times over a short period.

- In the object library, temporary entries are placed together following the permanent entries. Any time a permanent entry is added to the library, all temporary entries are deleted. Temporary entries are also deleted when you replace one permanent entry with another.

- In the source library, temporary and permanent entries can be in any order. One entry is placed after another regardless of their designations. Temporary entries, therefore, are not automatically deleted every time you add a permanent entry. However, when the source library is reallocated or reorganized, only permanent entries remain.

- You can use temporary entries as often as you like until they are deleted.

- A temporary entry cannot replace a permanent entry.

Permanent Entries

- Permanent entries are entries you intend to keep in your libraries. They are normally entries you use often or at regular intervals (once a week, once a month, and so on).

- The program will not delete permanent entries unless you use the delete function of the library maintenance program to delete them, or the allocate function to delete the entire library.

SIMULATION AREA VERIFICATION

The optional PACKIN (PACKIN-name) and/or PACKO (PACKO-name) parameters verify that the correct libraries are being accessed before the program performs the copy function. If the PACKIN parameter is supplied, the program compares the name given with the name in the volume label on the simulation area specified by the FROM parameter to ensure that they match. If the PACKO parameter is supplied, the program compares the name given with the name in the volume label on the simulation area specified by the TO parameter to ensure that they match. If not supplied, no verification is done.

The name specified in the PACKIN and/or PACKO parameters can be any combination of standard System/3 characters except apostrophes, leading or embedded blanks, and embedded commas. Its length must not exceed 6 characters. See Appendix A for a list of standard System/3 characters.

USING THE COPY FUNCTION

Reader-to-Library

Input

The program reads one library entry. It can be any one of the following types:

- Source statements
- Procedure
- Object program
- Routine

The entry is read from the system input device.

The header record on an object deck (H in column 1) contains the date the deck was punched. This date is in columns 58–63 and is in the format of the system date, either mmddyy or ddmmyy.

Output

- Duplicate characters are removed from source statements and procedures before they are put in the source library. The program does not check them for errors.
- Object programs and routines are placed in the object library after sequence and checksum information is removed.

Adding Entries

- The program can add a new entry to a library. The name of the entry is taken from the NAME parameter. See *Naming Library Entries* for valid names. The RETAIN parameter specifies whether the entry will be temporary or permanent. If the RETAIN parameter is omitted, RETAIN-T is assumed. (See *Retain Types.*)

Replacing Existing Entries

- The program can replace an existing library entry with the entry you are putting in the library. The RETAIN parameter specifies the new retain type. If the RETAIN parameter is omitted, RETAIN-T is assumed. A temporary entry cannot replace a permanent entry.
- The program can issue a message before replacing an existing entry. Whether it does depends on the RETAIN parameter you use. (See *RETAIN Parameter.*)
- Before the new entry is added, the duplicate entry is deleted. Additional library space is not needed unless the new entry is larger than the old one.

File-to-Library

Input

The disk file can contain one or more library entries. The entries must be in the format put out by the library-to-card function or by the linkage editor. The COPY statement at the beginning of each entry contains the name of the entry and the type of library (S, P, O, R). A CEND statement must follow each entry in the file.

The disk file must be a sequential file and be defined by a FILE statement in the OCL for the library maintenance program. Multivolume files are not supported.

Output

The output from the file-to-library function is the same as for the reader-to-library function.

Library-to-File

Input

The program can copy one or more library entries from a library to a disk file. The types of entries can be:

- Source statements
- Procedures
- Object programs
- Routines
- All of the preceding types

The NAME and LIBRARY parameters on the ENTRY statements specify which entries to copy.

Output

The output from the library-to-file function is of the same format as that from the library-to-card function. The output is written to a sequential disk file defined by an OCL FILE statement and created by the library maintenance program. Multivolume files are not supported.

Library-to-Library

Input

The program can copy one or more library entries from one simulation area to another. The types of entries can be:

- Source statements
- Procedures
- Object programs
- Routines
- All of the preceding types
- Minimum system

The NAME and LIBRARY parameters specify which entries to copy.

Output

- The entries, regardless of their type, are copied from one simulation area to another without change.
- Entries can be copied and renamed on the same simulation area by use of the NEWNAME parameter. (See *NEWNAME Parameter* and *Naming Library Entries*.)
- Copying a minimum system (LIBRARY-O, NAME-SYSTEM), or all of the types (LIBRARY-ALL, NAME-ALL), creates a system pack from which an IPL can be performed. (Copying LIBRARY-ALL, NAME-ALL will create a system pack only if the simulation area specified by the FROM parameter is a system pack.) The object library on the simulation area you specify in the TO parameter must be empty; that is, it cannot contain any entries or deleted entries. Also, the object library on the simulation area specified by the TO parameter must have been allocated with a scheduler work area and a checkpoint/restart area at least as large as those on the simulation area specified by the FROM parameter.
- The RETAIN parameter specifies whether the entries will be temporary or permanent. If the RETAIN parameter is omitted, RETAIN-T is assumed. When the parameters LIBRARY-ALL and NAME-ALL or LIBRARY-O and NAME-SYSTEM are used, RETAIN-P is assumed and RETAIN-T is invalid.

Adding Entries

- You can omit the NEWNAME parameter. If you do, the name used for the copy is taken from the NAME parameter. (The copy will have the same name as the original entry.)
- If NAME-ALL is specified, the names by which the entries are identified on the simulation area specified by the FROM parameter are also used on the simulation area specified by the TO parameter to identify the entries.

Replacing Existing Entries

- The program can replace existing entries with the entries you are putting in the library. If the entry you are copying (the entry on the simulation area you identify in the FROM parameter) has the same name as the entry you are replacing (the entry on the simulation area you identify in the TO parameter), you must omit the NEWNAME parameter because the NEWNAME parameter cannot be the same as the NAME parameter. If the names are not the same, you must use the NEWNAME parameter to give the name of the entry being replaced.
- The program can halt before replacing an existing entry. Whether it does depends on the RETAIN parameter. (See *RETAIN Parameter.*)
- A temporary entry cannot replace a permanent entry.

Library-to-Printer and/or Card

Types of Entries that Can Be Printed or Punched

The program can print or punch one or more library entries. They can be any one of the following types:

- Source statements
- Procedures
- Object programs
- Routines
- All of the preceding types (limited to entries having the same name and entries beginning with the same characters)

The program can print (but not punch) the following types of directory entries:

- Source statements
- Procedures
- Object programs
- Routines
- System directory
- All of the preceding types

The program will sort directory names before printing them only if there is enough work space to contain the directory on the simulation area specified by the FROM parameter.

Printed or Punched Library Entries

- Duplicate characters are reinserted into source statements and procedures to make them readable.
- Object programs and routines are printed and punched after sequence information and checksum information (punch only) has been added.
- The library entries, when punched, are preceded by a COPY statement of the reader-to-library format and followed by a CEND statement.

Printout of Directory Entries

- The format of the source library directory printout is described in Figure 4-73. If there is no source library, the NO SOURCE LIBRARY EXISTS message is logged. If a source library exists but is empty, the NO-SOURCE DIR ENTRIES EXIST message is logged.

- The format of the object library directory printout is described in Figure 4-74. If there is no object library, the NO OBJECT LIBRARY EXISTS message is logged. If an object library exists but is empty, the NO OBJECT DIR ENTRIES EXIST message is logged.

- A sample system directory printout is described in Figure 4-75. If there is no source library, the NO SOURCE LIBRARY EXISTS ON THIS PACK message is logged. If there is no object library, the NO OBJECT LIBRARY EXISTS ON THIS PACK message is logged.

The following fields in the directory printout are not updated when an object library entry is deleted:

- The number of available directory entries
- The location of the next available directory entry
- The next available library sector
- The number of available library sectors

These represent only contiguous space that can be used; therefore gaps are not included. In the object library section of the system directory printout the lines UNUSED SPACE FROM DELETED PERMANENTS and UNUSED SPACE FROM DELETED TEMPORARIES describe the space occupied by gaps. This space could be made available for use if the object library were to be reorganized. If this space is excessive, the library should be reorganized.

The final line of the system directory printout shows how a control statement might have been coded to result in the indicated allocation.

Library information is given in both sectors and tracks. When the number of active or available sectors is not evenly divisible by 24, a rounding scheme is used to determine the size in tracks. The fields describing active library space are rounded up when converting from sectors to tracks, and the fields describing available library space are rounded down in the conversion. For example, in the source library section, the lines DIRECTORY SPACE, PERMANENT LIBRARY SPACE, and ACTIVE LIBRARY SPACE are all referring to active space. Thus the number of tracks has been rounded up whenever a fraction of a track was active. The number of tracks representing AVAILABLE LIBRARY SPACE was rounded down in the conversion from sector size to track size.

PRINTOUT

```
SOURCE DIRECTORY FROM XX VOL. ID XXXXXX MM/DD/YY HH.MM.SS[1]

                        ADDRESS
TYPE     NAME    FIRST@      LAST@       ATTRI     #SECTORS
  X     XXXXXX  XXX-XX      XXX-XX        X         XXXX
```

Explanation:

Heading	Meaning
TYPE	S = source statements P = procedure
NAME	Name of library entry (up to six characters)
ADDRESS (FIRST and LAST)	Addresses of first and last sectors that contain the library entry. Addresses are expressed by track and sector numbers. Example: 008-03 means track 8, sector 3.
ATTRI	T = temporary P = permanent
#SECTORS	Total number of sectors used for the library entry.

[1] The time specified by HH.MM.SS is included in the printout only if the time-of-day function was selected during system generation.

Figure 4-73. Source Library Directory Printout

PRINTOUT

```
OBJECT DIRECTORY FROM XX VOL. ID XXXXXX MM/DD/YY HH.MM.SS[1]

                DISK    CYL/   TXT-  LINK   EXT  RLD  ENTRY  CORE                    TOT
TYPE   NAME     ADD     SEC    CAT   ADDR   BUF  DISP PNT    SEC   ATTR  LEVEL  SEC   CDATE-CTIME
XX X   XXXXXX   TTT/SS  CC/SS  XXX   XXXX   XXX  XX   XXXX   XXX   XXXX  XXX    XXXX  XX/XX HH.MM
```

Explanation:

Heading	Meaning
TYPE	The leftmost character printed indicates the attributes of the entry as follows: P = permanent T = temporary The middle character printed indicates the module class number as follows: Blank = class 0 1 = class 1 2 = class 2 3 = class 3 Class numbers are significant when jobs are being placed on the spool reader queue by $QCOPY under CCP and $QCOPY user authorization is required. A user may not execute a program with a class number higher than that for which he is authorized. (See *Program Classification* under $QCOPY.) The rightmost character printed indicates the type of module. Its meaning is as follows: O = object module R = routine
NAME	Name of library entry (up to 6 characters)
DSK ADD	Address where library entry begins on disk. Example: 015/10 means track 15, sector 10 (in decimal). T = track, S = sector.
CYL/SEC	Address where library entry begins on disk (in hexadecimal). C = cylinder, S = sector.

[1] The time specified by HH.MM.SS is included in the printout only if the time-of-day function was selected during system generation.

Figure 4-74 (Part 1 of 3). Object Library Directory Printout

Heading	Meaning
TXT-CAT	For object program, this number indicates the number of sectors used for the text portion of the library entry. Object programs consist of two parts: text and RLD. Text is the program; RLD is information used in loading the program for execution. For routines, this number is the category of the routine. This number is used by the overlay linkage editor for determining overlay structures.
LINK ADDR	Object programs only. Assigned hexadecimal link-edit address of this library entry. If the program has external buffers (attribute byte 1, bit 2=1 and attribute byte 2, bit 1=1), the low-order byte of the link-edit address is assumed to be X'00'.
EXT BUF	If the program has external buffers (attribute byte 1, bit 2=1 and attribute byte 2, bit 1=1), external buffer size is indicated in sectors.
RLD DISP	Object programs only. It indicates the hexadecimal position in which RLD information begins in the last text sector. If the last text sector contains no RLD information, the RLD displacement is 0, indicating that the information starts in the next sector.
ENTRY PNT	Object programs only. Main storage address (hexadecimal) where program execution begins before relocations.
CORE SEC	Main storage size, given in sectors, required to run the program.
ATTR	**Byte 1:** Bit 0=1 Permanent entry 0 Temporary entry Bit 1=1 Inquiry. This program requires that the Inquiry key be pressed to start processing. Bit 2=0 Must be zero if attribute byte 2 bit 1=0. 1 External buffers (if attribute byte 2, bit 1=1). This program uses disk data buffers that are outside the user program area but within the partition (see note 1). Bit 3 Program class (see note 2). Bit 4=1 Source required. This program requires the allocation of the $WORK and $SOURCE files. $SOURCE must be filled either from the system input device or from a source library. This program can be preceded by the macro processor. If a // SWITCH statement containing 1XXXXXXX was processed, the $SOURCE file is opened as input instead of output.

Notes:

1. A checkpoint/restart program will have attributes of hex 0040 and a program with external buffers will have attributes of hex 2040. The checkpoint/restart and external buffers attributes are mutually exclusive.
2. Attribute byte 1 bit 3 is the leftmost bit and attribute byte 2 bit 6 is the rightmost bit of a 2-bit binary program class number.

Figure 4-74 (Part 2 of 3). Object Library Directory Printout

Heading	Meaning	
ATTR (Continued)	Bit 5=1	Deferred mount. This program accepts mounting of data modules during its execution.
	Bit 6=1	PTF applied. A program temporary fix (PTF) has been applied to this program.
	Bit 7=1	Overlay object program
	Byte 2:	
	Bit 0=1	System input dedication. The system input device must be dedicated to this program. The device may be released when no longer needed.
	Bit 1=1	Checkpoint/restart program (if attribute byte 1 bit 2=0). (See note 1.) External buffers (if attribute byte 1, bit 2=1). This program uses disk data buffers that are outside the user program area but within the partition (see note 1).
	Bit 2=1	Direct source read. This program can have a // COMPILE statement and a no source required attribute (byte 1, bit 4=0). The program will access the source itself.
	Bit 3=1	This program has been link-edited or compiled using System Control Program Number 5704-SC2.
	Bit 4=1	Privileged program.
	Bit 5=1	Program common. This program requires that a new load address be calculated at load time to place it in main storage beyond its own program common region.
	Bit 6	Program class (see note 2).
	Bit 7=1	Memory resident overlay program.
LEVEL	Release level of system programs. For user programs, this level is a number assigned by the overlay linkage editor. For a COBOL, FORTRAN, or RPG II object program, the level is printed as COB, FOR, or RPG.	
TOT SEC	Total number of disk sectors occupied by the library entry.	
CDATE-CTIME	This heading is for COBOL, FORTRAN, and RPG II object programs; the date and time are printed only if the object program is compiled with release 3 or later of program number 5704-SC2. If CDATE-YES was specified on the COPY control statement, this is the date and time during which the object program was compiled; the same date and time appeared on the heading of the compile listing (time is shown only if timer support was generated on the system doing the compilation). The format of CDATE can be either mm/dd (month and day) or dd/mm (day and month) depending on the format selected for the system date during system generation. The format for CTIME is always hh.mm (hour.minute). If the CDATE parameter was not specified, the CDATE-NO parameter was specified, or the entry is for an R-module, there is no output for this heading.	

Notes:

1. A checkpoint/restart program will have attributes of hex 0040 and a program with external buffers will have attributes of hex 2040. The checkpoint/restart and external buffers attributes are mutually exclusive.
2. Attribute byte 1 bit 3 is the leftmost bit and attribute byte 2 bit 6 is the rightmost bit of a 2-bit binary program class number.

Figure 4-74 (Part 3 of 3). Object Library Directory Printout

```
                SYSTEM DIRECTORY FROM F1 VOLUME ID F1F1F1 05/03/76 00.04.19 (1)

                                --LOCATION--   ------SIZE------   --NUMBER--
LIBRARY AREA OVERVIEW           DISK ADDRESS   TRACKS   SECTORS   OF ENTRIES

SOURCE LIBRARY ALLOCATED SIZE                    50      1200
OBJECT DIRECTORY ALLOCATED SIZE                   5       120        1439
OBJECT LIBRARY ALLOCATED SIZE                   300      7200
START OF LIBRARIES              008-00
SYSTEM HISTORY AREA             058-00           4
CHECKPOINT/RESTART AREA         062-00          15
SCHEDULER WORK AREA             077-00          18
END OF LIBRARIES                394-23


        SOURCE LIBRARY SECTION

START OF SOURCE LIBRARY         008-00
  DIRECTORY SPACE                                1         8
  PERMANENT LIBRARY SPACE                       48      1131
  ACTIVE LIBRARY SPACE                          48      1131
  AVAILABLE LIBRARY SPACE       055-11           2        61
END OF SOURCE LIBRARY           057-23


        OBJECT LIBRARY SECTION

START OF OBJECT DIRECTORY       095-00
  AVAILABLE PERMANENT DIRECTORY ENTRIES  098-17-147                      364
  AVAILABLE TEMPORARY DIRECTORY ENTRIES  098-18-042                      357
END OF OBJECT DIRECTORY         099-23
START OF OBJECT LIBRARY         100-00
  AVAILABLE PERMANENT LIBRARY SPACE     389-20     5       124
  UNUSED SPACE FROM DELETED PERMANENTS                6       150
  AVAILABLE TEMPORARY LIBRARY SPACE     390-22       4        98
  UNUSED SPACE FROM DELETED .TEMPORARIES              0         9
  ACTIVE LIBRARY SPACE                              285      6823
  PERMANENT OBJECT LIBRARY SPACE                    228      5469
  PERMANENT ROUTINE LIBRARY SPACE                    56      1337
ALLOCATED END OF OBJECT LIBRARY 394-23
EXTENDED END OF OBJECT LIBRARY  394-23


SOURCE-50,DIRSIZE-5,OBJECT-300,HISTORY-4,SYSTEM-YES
```

(1) The time specified by HH.MM.SS is included in the printout only if the time-of-day function was selected during system generation.

Figure 4-75. System Directory Printout

$MAINT—Delete Function

USES

- Delete a temporary or permanent entry from a library (or entries with the same name from all libraries).
- Delete temporary or permanent library entries that have names beginning with certain characters.
- Delete all temporary or permanent library entries of a certain type.

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

All volumes referenced by the control statements must remain online during the execution of $MAINT.

Delete a Temporary or Permanent Library Entry (or Entries with the Same Name from All Libraries)

// DELETE FROM-code,LIBRARY-{S|P|O|R|ALL},NAME-name [,RETAIN-{T|P}] [,PACKIN-name]

Delete Temporary or Permanent Entries with Names Beginning with Certain Characters

// DELETE FROM-code,LIBRARY-{S|P|O|R|ALL},NAME-characters.ALL [,RETAIN-{T|P}] [,PACKIN-name]

Delete All Temporary or Permanent Entries of a Certain Type

// DELETE FROM-code,LIBRARY-{S|P|O|R},NAME-ALL [,RETAIN-{T|P}] [,PACKIN-name]

CONSIDERATIONS AND RESTRICTIONS

- System modules cannot be deleted from the active system pack (the simulation area the system was loaded from at IPL time).

- Library maintenance program modules cannot be deleted from the active program pack for the partition in which the library maintenance program was loaded.

- When all temporary entries are deleted from the object library using LIBRARY-O, NAME-ALL, RETAIN-T, the temporary routines (LIBRARY-R) are also deleted.

- The RETAIN parameter must match the attribute of the entry in the library; otherwise the entry is considered not found. RETAIN-T is assumed if the RETAIN parameter is omitted.

- The optional PACKIN parameter (PACKIN-name) verifies that the correct library is being accessed before the program performs the delete function. If this parameter is not supplied, no verification is done.

PARAMETER SUMMARY

Parameter	Description
FROM-code	The simulation area that contains library entries you are deleting. Possible codes are R1, F1, R2, F2.
LIBRARY-{S / P / O / R / ALL}	Type of entries being deleted. Possible codes are: **Code** — **Meaning** S — Source statements (source library) P — Procedures (source library) O — Object programs (object library) R — Routines (object library) ALL — All types of entries (S, P, O, and R) are being deleted
NAME-{name / characters.ALL / ALL}	Particular entries, of the type indicated in the LIBRARY parameter, being deleted. These entries are further identified by the RETAIN parameter. Possible codes are: **Code** — **Meaning** name — Name of the library entry, or entries, being deleted. characters.ALL — Entries that have names beginning with the indicated characters. You can use up to 5 characters. Example: NAME-INV.ALL refers to the entries having names that begin with INV. ALL — All entries (of the type indicated in LIBRARY parameter). NAME-ALL cannot be used with LIBRARY-ALL.
RETAIN-{T / P}	Designation of entries being deleted: **Code** — **Meaning** T — Temporary P — Permanent
PACKIN-name	Name of the simulation area specified by the FROM parameter. The name can be any combination of standard System/3 characters except apostrophes, leading or embedded blanks, and embedded commas. Its length must not exceed 6 characters. See Appendix A for a list of standard System/3 characters.

$MAINT—Modify Function

USES

- Maintain source statements and procedures by using a card reader.
- Reserialize a source library entry.
- List the statements in a source library entry.
- Remove statements from a source library entry.
- Replace source library statements.
- Insert statements into a source library entry.

CONSIDERATIONS AND RESTRICTIONS

- Sequence numbers are a physical part of the source record and must be placed where they will not conflict with other data in the record. In a procedure they should be placed near the end of the record beyond the OCL and control statements' keywords and parameters. The sequence numbers should be placed in source statements where they will not overlay data. For example, data could be destroyed if sequence numbers were placed in RPG II source statements that contained compile-time tables.
- At least three control statements must be entered to modify the source library. A MODIFY statement is needed to describe the library entry. A REMOVE, REPLACE, or INSERT statement describes the type of modification. A CEND statement indicates the end of the MODIFY control statements.
- The simulation area specified by the WORK parameter on the MODIFY statement must contain a work area large enough to hold the modified source library entry.
- The sequence numbers specified by the FROM-seqno, TO-seqno, and AFTER-seqno parameters on the REMOVE, REPLACE, or INSERT statements must be valid numbers and exist in the source library entry. There are no default values for these parameters. The number of digits entered must be the same as the number of positions specified by the SEQFLD parameter.
- All statements in a source library entry must have ascending sequence numbers in the positions specified by the SEQFLD parameter.
- Multiple operations (REMOVE, REPLACE, INSERT) may be performed within the same MODIFY run if they are done in an ascending sequential order. That is, the FROM sequence number in a REMOVE or REPLACE statement must be greater than the last sequence number in the preceding statement. The AFTER sequence number of an INSERT statement must be equal to or greater than the last sequence number of the preceding statement. Consecutive INSERT statements must not have the same sequence number.
- When modification is complete, the directory entry is written back with a permanent attribute.
- The control statements following the MODIFY statement are read from the system input device.
- Since the REMOVE control statement is valid for both the $DELET program and $MAINT program, care should be used when a $DELET procedure is modified. The $MAINT program will attempt to determine if the REMOVE statement is a data record or a control statement. If a determination cannot be made, the program will halt and wait for the operator's instructions.
- If LIST-YES is specified and a printer error (causing a system message) occurs during the listing of the source library entry, responding to the message with a 2 option causes the listing to stop. The modified entry will be placed back in the library before the function is terminated with the controlled cancel.
- The optional PACKIN parameter (PACKIN-name) verifies that the correct library is being accessed before the program performs the modify function. If not supplied, no verification is done.

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

All volumes referenced by the control statements must remain online during the execution of $MAINT.

Initiate Modification

// MODIFY NAME-name,FROM-code,LIBRARY-{S|P},WORK-code [,RESER-{YES|NO|ONLY}] [,LIST-{YES|NO}]
[,SEQFLD-xxyy] [,INCR-number] [,PACKIN-name]

Control Statements Following // MODIFY

Delete all statements between and including the FROM and TO sequence numbers.

// REMOVE FROM-seqno, TO-seqno

Replace all statements between and including the FROM and TO sequence numbers with the statements supplied.

// REPLACE FROM-seqno, TO-seqno
-
-
1 - n statements to replace those removed
-

Insert the supplied statements after the statement indicated by the AFTER parameter.

// INSERT AFTER-seqno
-
1 - n statements to be inserted

// CEND ◢ Must follow the control statements to terminate the modify function.

PARAMETER SUMMARY

Parameter	Description
NAME-name	Name of the entry you are modifying. This is the name that identifies the entry in the library directory.
FROM-code	Simulation area that contains the entry you are modifying. Possible codes are R1, F1, R2, F2.
LIBRARY-{S / P}	Type of library entry you are modifying. Possible codes are: **Code** — **Meaning** S — Source statements (source library) P — Procedures (source library)
WORK-code	The simulation area containing space the program can use as a work area. Possible codes are R1, F1, R2, F2.
RESER-{YES / NO / ONLY}	Specifies whether reserialization should be performed when the entry is placed back in the source library. When reserialization is specified, the first record in the entry is assigned record number 0. Possible information is: **Information** — **Meaning** YES — The entry is reserialized. NO — The entry is not reserialized. NO is assumed if the RESER parameter is omitted. ONLY — Reserialize only; no other maintenance is performed. When this is coded, no REMOVE, REPLACE, INSERT, or CEND statements can be entered.
LIST-{YES / NO}	Specifies whether the source library entry should be listed as the modified entry is placed back in the source library. NO is assumed if the LIST parameter is omitted.
SEQFLD-xxyy	The starting and ending positions of the field that contains the sequence number. The sequence number can be up to eight digits long. The starting position is entered first (xx) and then the ending position (yy). If this parameter is not entered, 9296 is assumed.
INCR-number	Increment value for the sequence field if reserialization (RESER-YES or RESER-ONLY) is specified. The value can be up to five digits. If this parameter is not entered, a value of 10 is assumed.
PACKIN-name	Name of the simulation area specified by the FROM parameter. The name can be any combination of standard System/3 characters except apostrophes, leading or embedded blanks, and embedded commas. Its length must not exceed 6 characters. See Appendix A for a list of standard System/3 characters.

REMOVE, REPLACE, INSERT PARAMETERS

FROM-seqno	The sequence number of the first statement to be used in the operation.
TO-seqno	The sequence number of the last statement to be used in the operation.
AFTER-seqno	The sequence number of the statement after which the new statements are to be added.

$MAINT—Rename Function

USES

- Change the name of a library entry.
- Change the name of library entries that have names beginning with certain characters.

CONSIDERATIONS AND RESTRICTIONS

- System modules should not be renamed on the active system pack (the simulation area from which the system was loaded during IPL).
- Library maintenance modules should not be renamed on the active program pack for the partition in which the library maintenance program was loaded.
- The optional PACKIN parameter (PACKIN-name) verifies that the correct library is being accessed before the program performs the rename function. If this parameter is not supplied, no verification is done.

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

All volumes referenced by the control statements must remain online during the execution of $MAINT.

```
// RENAME FROM-code,LIBRARY-{S|P|O|R},NAME-name,NEWNAME-name[,PACKIN-name]

// RENAME FROM-code,LIBRARY-{S|P|O|R},NAME-characters.ALL,NEWNAME-characters[,PACKIN-name]
```

PARAMETER SUMMARY

Parameter	Description
FROM-code	The simulation area that contains the entry you are renaming. Possible codes are R1, F1, R2, F2.
LIBRARY-{S, P, O, R}	Type of library entry you are renaming. Possible codes are: **Code** — **Meaning** S — Source statements (source library) P — Procedures (source library) O — Object programs (object library) R — Routines (object library)
NAME-name	Current name of the entry you are renaming. This is the name that identifies the entry in the library directory.
NAME-characters.ALL	Only those entries beginning with the indicated characters. You can use up to 5 characters.
NEWNAME-name	New name you want to give the entry. Follow these rules to construct the name: 1. You can use any System/3 characters except blanks, commas, quotes, and periods (Appendix A lists the characters). The names of most IBM programs begin with a dollar sign ($). Therefore, to avoid possible duplication, do not use a dollar sign as the first character in the names you use for your entries. The first character must be alphabetic. 2. You can use up to 6 characters, but you cannot use the names ALL, DIR, and SYSTEM. They have special meanings in the NAME parameter.
NEWNAME-characters	Beginning characters you want to use in names identifying the copies. You can use use up to 5 characters.
PACKIN-name	Name of the simulation area specified by the FROM parameter. The name can be any combination of standard System/3 characters except apostrophes, leading or embedded blanks, and embedded commas. Its length must not exceed 6 characters. See Appendix A for a list of standard System/3 characters.

OCL CONSIDERATIONS

The following OCL statements are needed to load the library maintenance program.

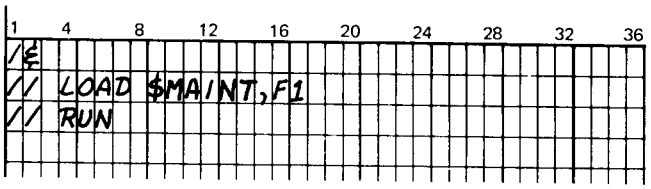
```
// LOAD $MAINT,code
// RUN
```

The code you supply depends on the location of the simulation area containing the library maintenance program. Possible codes are R1, F1, R2, F2.

If the copy file-to-library or library-to-file functions are to be used in this run of the $MAINT program, the necessary disk FILE OCL statements must follow the LOAD statement and precede the RUN statement.

$MAINT—Examples

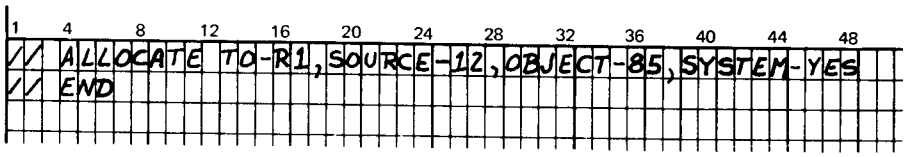
Figures 4-76 through 4-93 illustrate the functions of the library maintenance program. Figure 4-76 is an example of the OCL needed to load the program. The other figures are examples of the control statements necessary to carry out the specified functions.



Explanation:

- The library maintenance program is loaded from F1.

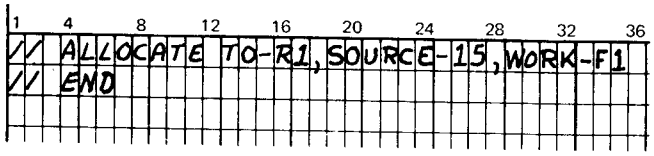
Figure 4-76. OCL Load Sequence for Library Maintenance



Explanation:

- Libraries are being created on R1 (TO-R1 in ALLOCATE statement).
- Source library space is 12 tracks (SOURCE-12).
- Object library space is 85 tracks (OBJECT-85). The object library will contain system programs (SYSTEM-YES). Thus, the disk area will also include space for the scheduler work area. The system history area size will default to 2 tracks.
- The directory will be 3 tracks.

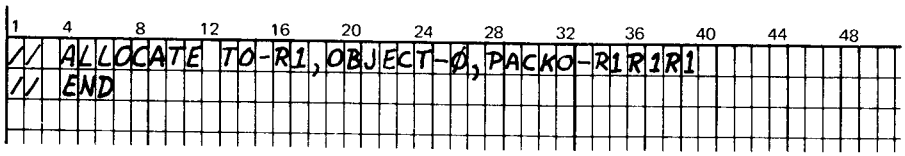
Figure 4-77. Allocate Example: Creating Both Source and Object Libraries on a Disk



Explanation:

- The source library is located on R1 (TO-R1 in ALLOCATE statement).
- The size of the source library is being changed to 15 tracks (SOURCE-15).
- Any time the program changes the size of a library, it reorganizes the library. To do this, it needs a work area. This area is on F1 (WORK-F1).

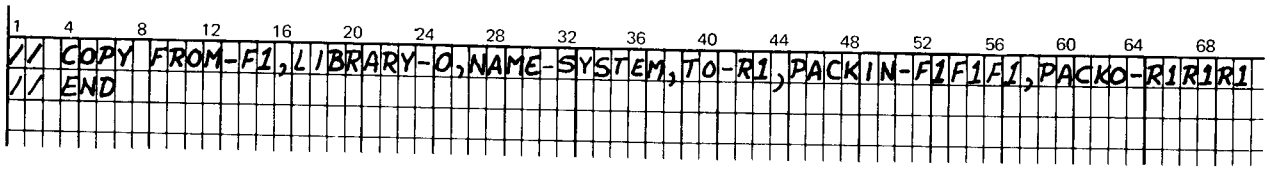
Figure 4-78. Allocate Example: Changing the Size of a Source Library



Explanation:

- The object library is located on R1 (TO-R1 in ALLOCATE statement).
- The OBJECT-0 parameter tells the program to delete the object library. If a scheduler work area and system history area precede the object library, they are also deleted.
- The library maintenance program verifies that the simulation area name is R1R1R1 before the allocate function is performed (PACKO-R1R1R1).

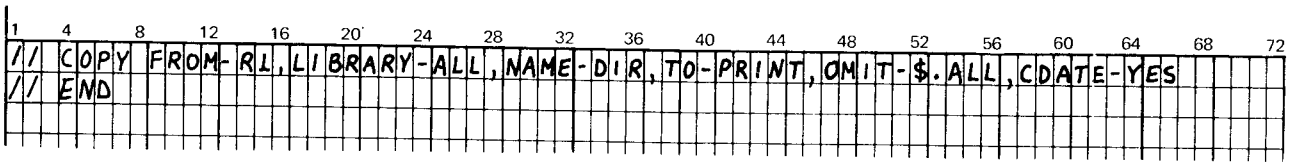
Figure 4-79. Allocate Example: Deleting the Object Library from a Disk



Explanation:

- System programs are in the object library on F1 (LIBRARY-O and FROM-F1 in COPY statement).
- The NAME parameter (NAME-SYSTEM) tells the program to copy the system programs.
- The simulation area that is to contain the copy is R1 (TO-R1).
- The library maintenance program verifies that the simulation area name of the FROM unit is F1F1F1 (PACKIN-F1F1F1) and that the simulation area name of the TO unit is R1R1R1 (PACKO-R1R1R1) before the copy function is performed.

Figure 4-80. Copy Example: Copying Minimum System from One Disk to Another

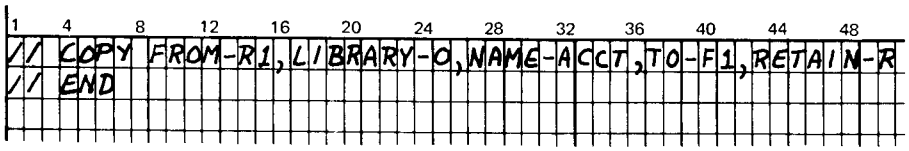


Explanation:

All library directories and the system directory on R1 are printed (COPY statement):

- FROM identifies the simulation area containing the directories.
- LIBRARY indicates which directories are to be printed.
- NAME and TO indicate that the program is to be printing directories.
- Entries beginning with a $ are not printed.
- Print the date and time of compilation for any COBOL, FORTRAN, or RPG user programs in the object library (CDATE-YES).

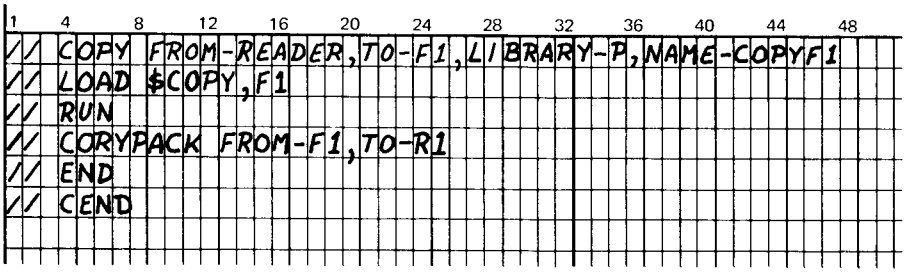
Figure 4-81. Copy Example: Printing Library Directories



Explanation:

- LIBRARY-O, NAME-ACCT, and FROM-R1 in the COPY statement tell the program to read the object program named ACCT from R1.
- TO-F1 tells the program to copy the object program to F1. There is no NEWNAME parameter in the COPY statement. Therefore, the name the program will have on F1 is ACCT (NAME-ACCT). Since the old version of the program already exists on F1 under that name, the old version is replaced.
- The library maintenance program normally issues a message before replacing a library entry. The RETAIN-R parameter, however, tells the program to omit that message.

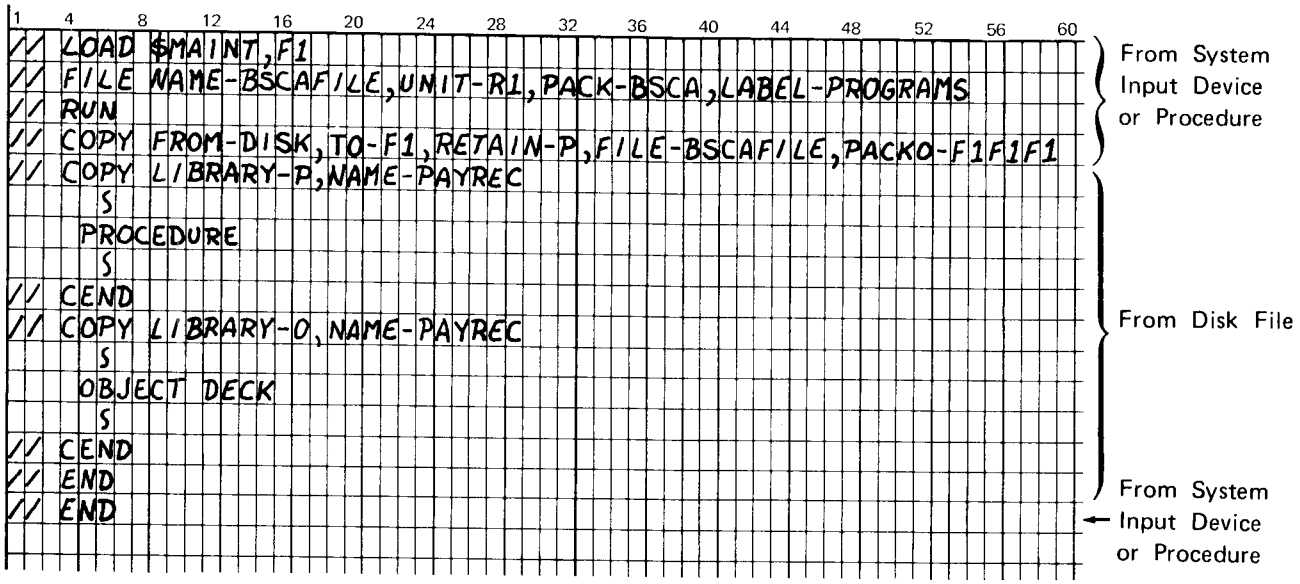
Figure 4-82. Copy Example: Copying Object Program to F1



Explanation:

- FROM-Reader in the COPY statement tells the library maintenance program to read the module from the system input device.
- The procedure (LIBRARY-P) will be written to the source library on F1 (TO-F1), named COPYF1 (NAME-COPYF1), and given the default attribute of temporary.
- All statements are entered into the library until the CEND statement is read to terminate the copy.

Figure 4-83. Copy Example: Copying Procedure from System Input Device



Explanation:

- The OCL for a file-to-library copy must contain a FILE statement for the disk file.
- The filename on the COPY statement (FILE-BSCAFILE) matches the filename on the OCL FILE statement (NAME-BSCAFILE).
- The COPY statement does not contain an RECL parameter, so a record length of 96 is assumed.
- The library maintenance program verifies that the simulation area name of the TO unit is F1F1F1 before the copy function is performed (PACKO-F1F1F1).
- All source and object decks in the disk file must have a // COPY statement as the first card image and a // CEND statement as the last card image. When logged, the // is changed to XX to indicate that they were read from disk rather than from the system input device or procedure.
- The // END statement read from the file (printed XX END), causes the next statement to be read from the system input device or procedure. An END statement must still be read from the system input device or procedure to indicate the end of the library maintenance control statements.

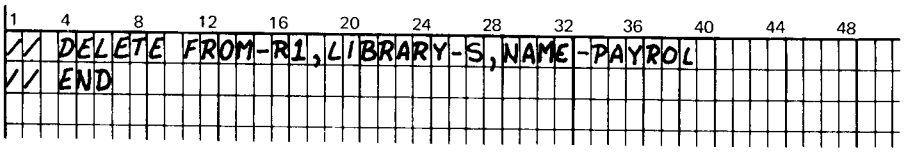
Figure 4-84. Copy Example: Disk File to Library

```
1   4    8    12   16   20   24   28   32   36   40   44   48   52   56   60   64
// LOAD $MAINT,F1
// FILE NAME-BACKUP,UNIT-D1,PACK-D1D1D1,LOCATION-2Ø/Ø,TRACKS-8Ø
// RUN
// COPY FROM-R1,TO-DISK,RECL-8Ø,FILE-BACKUP,PACKIN-R1R1R1
// ENTRY LIBRARY-ALL,NAME-PAY.ALL
// ENTRY LIBRARY-S,NAME-ALL
// ENTRY LIBRARY-O,NAME-INVENT
// NEND
// END
```

Explanation:

- The OCL for a library-to-file copy must contain a FILE statement for the disk file.
- The filename on the COPY statement (FILE-BACKUP) matches the filename on the OCL FILE statement (NAME-BACKUP).
- The library maintenance program verifies that the simulation area name of the FROM unit is R1R1R1 before the copy function is performed (PACKIN-R1R1R1).
- A sequential file with record length of 80 (RECL-80) will be created on D1.
- The file will contain entries from all libraries with names beginning with the characters PAY, all source library entries, and object entry INVENT.
- The copy to file BACKUP is terminated by the NEND statement.
- The END statement indicates the end of the library maintenance control statements.

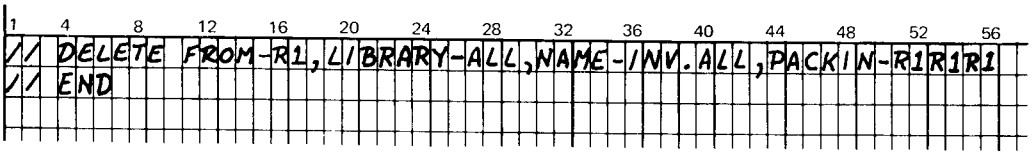
Figure 4-85. Copy Example: Library to Disk File



Explanation:

- The program deletes a set of source statements (LIBRARY-S in DELETE statement) named PAYROL (NAME-PAYROL) from R1 (FROM-R1) that has a temporary attribute.

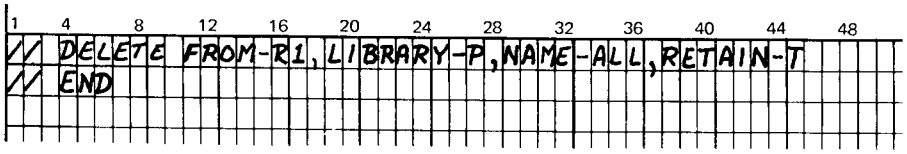
Figure 4-86. Delete Example: Deleting an Entry from a Library



Explanation:

- The entries being deleted are on R1 (FROM-R1 in DELETE statement).
- The library maintenance program verifies that the simulation area name is R1R1R1 before the delete function is performed (PACKIN-R1R1R1).
- The program deletes all entries from both source and object libraries (LIBRARY-ALL) that have names beginning with the characters INV (NAME-INV.ALL), with temporary attributes.

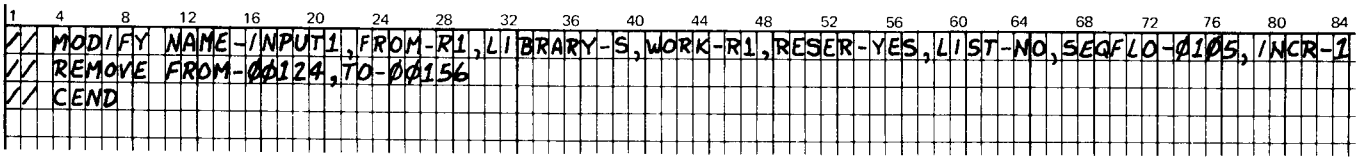
Figure 4-87. Delete Example: Deleting All Entries with Names that Begin with Certain Characters



Explanation:

- The entries being deleted are on R1 (FROM-R1 in DELETE statement).
- All temporary procedures are being deleted from the source library (LIBRARY-P,NAME-ALL).

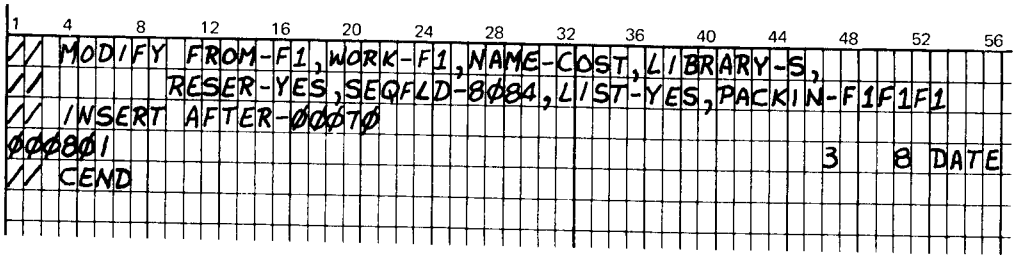
Figure 4-88. Delete Example: Deleting All Library Entries of One Type



Explanation:

- The source module named INPUT1 on R1 is being modified.
- The work space is on R1.
- The sequence numbers are in positions 1–5 of the statements.
- Sequence numbers 00124–00156 are being deleted from the module.
- The module is reserialized with increments of one.
- The module is not listed.

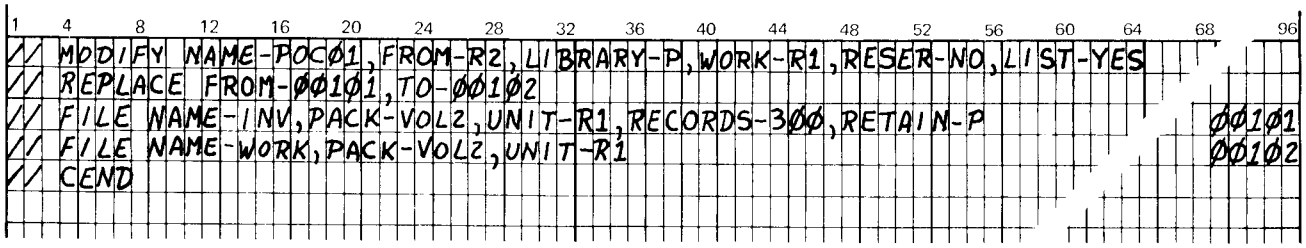
Figure 4-89. Modify Example: Removing Source Statements from a Module



Explanation:

- The source module COST on F1 is being modified.
- The library maintenance program verifies that the simulation area name of the FROM unit is F1F1F1 before the modify function is performed.
- The work space is on F1.
- The sequence numbers are in positions 80–84 of the statements.
- A statement is being inserted after statement number 00070.
- The module is reserialized with the default increment value of 10.
- The module is listed.

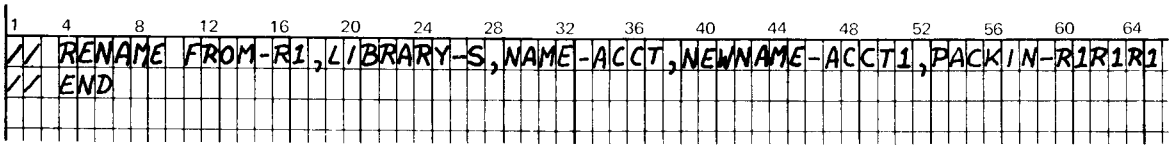
Figure 4-90. Modify Example: Inserting a Statement in a Source Module



Explanation:

- The procedure named POC01 on R2 is being modified.
- The work space is on R1.
- The sequence numbers are in default positions 92–96.
- Statements with sequence numbers 00101–00102 are being replaced.
- The module is not reserialized.
- The module is listed.

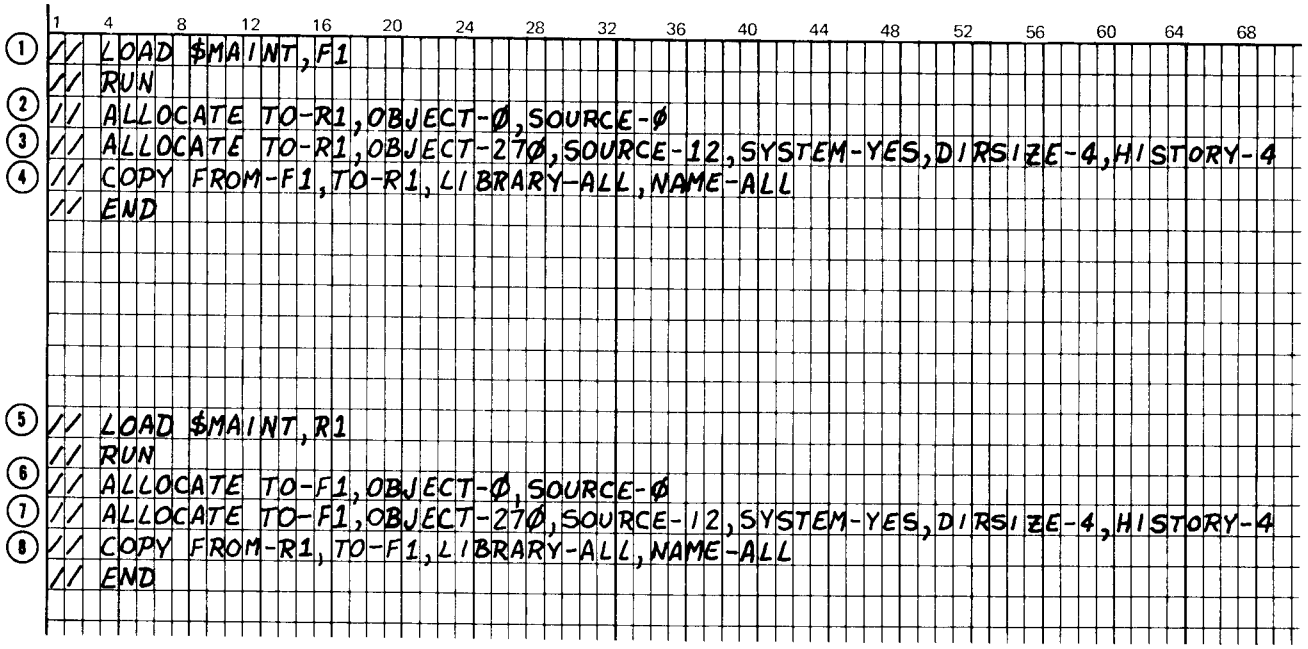
Figure 4-91. Modify Example: Replacing Statements in a Procedure



Explanation:

- R1 contains the entry being renamed (FROM-R1 in RENAME statement).
- The library maintenance program verifies that the simulation area name is R1R1R1 before the rename function is performed (PACKIN-R1R1R1).
- The entry is a set of source statements in the source library (LIBRARY-S). Its name is ACCT (NAME-ACCT).
- The entry name is being changed to ACCT1 (NEWNAME-ACCT1).

Figure 4-92. Rename Example: Renaming a Set of Source Statements in a Source Library



Explanation:

(1) $MAINT is loaded from F1.

(2) The libraries on R1 are deallocated (if present).

(3) New library space is allocated on R1.

(4) The libraries are copied from F1 to R1. The object library is reorganized as it is copied. Temporary entries become permanent when copied (see *Using the Copy Function, Library-to-Library, Output*).

(5) $MAINT is loaded from R1.

(6) The libraries on F1 are deallocated.

(7) New library space is allocated on F1.

(8) The libraries are copied back to F1. R1 could be used as backup. It contains the same libraries as F1.

Figure 4-93. Reorganizing the System Pack

Spool File Copy Program—$QCOPY

PROGRAM DESCRIPTION

The spool file copy program ($QCOPY) executes under control of the system control program (SCP) or the communications control program (CCP). The functions of $QCOPY supported by each control program are as follows:

- System control program (SCP)
  - Copy an entire spool file from: disk to disk, disk to tape, or tape to disk.
  - Copy all or part of a spool print queue, spool punch queue, or spool reader queue to: a device independent file or an IBM 1403 Printer (print queue only).
  - Copy to the spool reader queue from: a device independent file or the system input device.
  - Copy a display of the status of the spool queues to a device independent file.
  - Copy all or part of a spool queue from one spool file to another.
  - Restore (copy) the print or punch records (which were previously intercepted on the spool file and copied by $QCOPY to a disk or tape file) to the print or punch device or to the spool print or spool punch queue.
  - Maintain the authorization file.
  - Assign class numbers to programs.
- Communications control program (CCP)
  - Copy all or part of a spool print queue, spool punch queue, or spool reader queue to a disk file.
  - Copy to the spool reader queue from: a disk file, an IBM 3275 Display Station (Model 1 or 2), or an IBM 3277 Display Station (Model 1 or 2).
  - Copy a display of the status of the spool queues to: a disk file, an IBM 3275 Display Station (Model 1 or 2), or an IBM 3277 Display Station (Model 1 or 2).
  - Copy all or part of a spool queue from one spool file to another.
  - Restore (copy) the print or punch records (which were previously intercepted on the spool file and copied by $QCOPY to a disk file) to the spool print or spool punch queue.
  - Restrict the use of $QCOPY to only those functions for which individual users are authorized.

The spool file copy program can copy the active spool file as well as any inactive spool file.

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

Function	Control Statements
Copy the entire spool file	**1** // COPYSP FROM-{TAPE / code / SP},TO-{code / TAPE}
Copy selected job steps from the print queue	// COPYPRTQ [UNIT-code] [,FORMSNO-xxx] [,JOBN-{jobname / characters**} [,STEPN-{stepname / characters**}]] [,LENGTH-{FLR / VLR}] [,REMOVE-{NO / YES / ONLY}] [,OUTPUT-{ADD / FILE / PRINT}] [,FILE-{PRINTQ / filename}] [,HEADER-{NO / YES}] [,STOP-{NO / STEP / FORM}]
Copy selected job steps from the punch queue	// COPYPCHQ [UNIT-code] [,CARDNO-xxx] [,JOBN-{jobname / characters**} [,STEPN-{stepname / characters**}]] [,REMOVE-{NO / YES / ONLY}] [,OUTPUT-{ADD / FILE / PUNCH}] [,FILE-{PUNCHQ / filename}] [,HEADER-{NO / YES}] [,STOP-{NO / STEP / CARD}]

1 This control statement is valid only when the spool file copy program is executed under the system control program.

Function	Control Statements
Copy jobs to or from the reader queue	// COPYRDRQ [UNIT-code] [,RECL-nnn] [,INPUT-{FILE / READER / TERMINAL}] [,FILE-{READERQ / filename}] [,KEY-characters] [,LOKEY-characters] [,HIKEY-characters] [,LOREC-number] [,HIREC-number] [,OUTPUT-{FILE / ADD}] [,JOBN-{jobname / characters**}] [,REMOVE-{NO / YES / ONLY}] [,PARTITION-{1 / 2 / 3}]
Read $QCOPY control statements from a file	[1] // COPYCTRL [,FILE-{CONTROL / filename}]
Copy a display of the status of spool queues to a file	// DISPLAY [UNIT-code] [,OUTPUT-{ADD / FILE / TERMINAL}] [,FILE-{DISPLAYQ / filename}] [,Q-{RQ / PQ / WQ}]
Restore (copy) the print or punch queue records to the print or punch spool queue or device. These records were previously intercepted on the spool file and written by $QCOPY to a file.	// RESTORE [FILE-{RESTORE / filename}] [,JOBN-{jobname / characters**} [,STEPN-{stepname / characters**}]] [,{FORMSNO / CARDNO}-xxx] [,STOP-{NO / STEP / FORM / CARD}] [,OUTPUT-{SPOOL / PRINT / PUNCH}] [,UNIT-code]

[1] This control statement is valid only when the spool file copy program is executed under the communications control program.

Function	Control Statements
Copy selected job steps (by queue) from one spool file to another spool file.	// COPYQ Q-{RQ / WQ / PQ}, FROM-{code / SP}, TO-{code / SP} [,JOBN-{jobname / characters**} [,STEPN-{stepname / characters**}]] [,REMOVE-{NO / YES}] [,PARTITION-{1 / 2 / 3}] [,PRIORITY-{1 / 2 / 3 / 4 / 5}] [,{FORMSNO / CARDNO}-xxx]
Maintain the authorization file	[1] // AUTHORIZE [LIST-{NO / YES}]
Assign class numbers to programs	[1] // CLASSIFY PROGRAM-programname, UNIT-code, CLASS-{0 / 1 / 2 / 3} [,LIBRARY-{O / R}] [,PACK-packname]
	// END

[1] This control statement is valid only when the spool file copy program is executed in a batch partition.

PARAMETER SUMMARY

Parameter	Description
COPYSP Statement	
FROM-TAPE	Copies the spool file from a tape file. The tape unit must be specified in an OCL FILE statement with the parameter NAME-$SPOOL.
FROM-code	Specifies the location of the area containing the spool file to be copied. Possible codes are those for the main data areas.
FROM-SP	Copies the active spool file. This parameter is valid only when spool is active.
TO-TAPE	Copies the spool file to a tape file. The tape unit must be specified in an OCL FILE statement with the parameter NAME-$SPOOL.
TO-code	Specifies the location of the area to receive the copy of the spool file. Possible codes are those for the main data areas.
COPYPRTQ Statement	
UNIT-code	Identifies the location of the spool file containing the print queue to be copied. Possible codes are those for the main data areas. If this parameter is not specified, job steps are copied from the print queue on the active spool file.
FORMSNO-xxx	Copies only those job steps whose forms type matches the FORMSNO parameter.
JOBN-{jobname / characters**}	Copies only those job steps whose jobname either matches the JOBN parameter or begins with the same characters that precede the **.
STEPN-{stepname / characters**}	Copies only those job steps whose stepname either matches the STEPN parameter or begins with the same characters that precede the **. When STEPN is specified, JOBN must also be specified.
LENGTH-{FLR / VLR}	Specifies the format of the copy of the data. FLR specifies a fixed length record and VLR specifies a variable length record.
REMOVE-{NO / YES / ONLY}	Specifies whether the job step is to be only copied (REMOVE-NO), copied and removed from the print queue (REMOVE-YES), or removed without being copied (REMOVE-ONLY).
OUTPUT-{FILE / PRINT / ADD}	Specifies whether the output is to be copied to a file or printed. If copied to a file, the output can be added to the current logical end of the file rather than at the beginning of the file.
FILE-{PRINTQ / filename}	Specifies the name of the file to which the output is copied. A FILE statement defining this file must be included in the OCL when the spool file copy program is loaded.

Parameter	Description
COPYPRTQ Statement (continued)	
HEADER-{NO / YES}	Specifies whether a record containing information about the job step is to be placed in the output file immediately preceding the step.
STOP-{NO / STEP / FORM}	Specifies whether a message is to be issued which indicates the forms type and the number of pages required to print the next job step. The message is issued before each step is copied (STOP-STEP) or when the forms type changes (STOP-FORM). No message is issued if the parameter is STOP-NO (default).
COPYPCHQ Statement	
UNIT-code	Identifies the location of the spool file containing the punch queue to be copied. Possible codes are those for the main data areas. If this parameter is not specified, job steps are copied from the punch queue on the active spool file.
CARDNO-xxx	Copies only those job steps whose card type matches the CARDNO parameter.
JOBN-{jobname / characters**}	Copies only those job steps whose jobname either matches the JOBN parameter or begins with the same characters that precede the **.
STEPN-{stepname / characters**}	Copies only those job steps whose stepname either matches the STEPN parameter or begins with the same characters that precede the **. When STEPN is specified, JOBN must also be specified.
REMOVE-{NO / YES / ONLY}	Specifies whether the job step is to be only copied (REMOVE-NO), copied and removed from the punch queue (REMOVE-YES), or removed without being copied (REMOVED-ONLY).
OUTPUT-{FILE / PUNCH / ADD}	OUTPUT-FILE specifies that the output is to contain the control information (Q- and R-bytes) which is used by spool to punch the record. OUTPUT-PUNCH specifies that the output is not to contain the control information. The OUTPUT-ADD parameter causes the same operation as the OUTPUT-FILE parameter, except the output records are added to the current logical end of the file rather than at the beginning.
FILE-{PUNCHQ / filename}	Specifies the name of the file to which the output is copied. A FILE statement defining this file must be included in the OCL when the spool file copy program is loaded.
HEADER-{NO / YES}	Specifies whether a record containing information about the job step is to be placed in the output file immediately preceding the step.
STOP-{NO / STEP / CARD}	Specifies whether a message is to be issued which indicates the card type for the next job step to be copied. The message can be issued before each step is copied (STOP-STEP) or when the card type changes (STOP-CARD). No message is issued if the parameter is STOP-NO (default).

Parameter	Description
COPYRDRQ Statement	
UNIT-code	Identifies the location of the spool file containing the reader queue. Possible codes are those for the main data areas. If this parameter is not specified, jobs are copied to or from the reader queue on the active spool file.
RECL-nnn	Specifies the length of the spooled records on the reader queue when copying to the reader queue.
INPUT-{FILE / READER / TERMINAL}	Identifies the input record source when copying to the reader queue.
FILE-{READERQ / filename}	Specifies the name of the file from which jobs are copied to the spool reader queue or to which jobs are copied from the spool reader queue. A FILE statement defining this file must be included in the OCL when the spool file copy program is loaded.
KEY-characters	This optional parameter specifies that when copying to the spool reader queue from an indexed file, only those records whose keys begin with the characters specified on this parameter are to be copied. The number of characters that can be included on this parameter is from 1 through 29.
LOKEY-characters	This optional parameter specifies that when copying to the spool reader queue from an indexed file, only those records whose key is greater than or equal to the characters specified on this parameter are to be copied. The number of characters that can be included on this parameter is from 1 through 29.
HIKEY-characters	This optional parameter specifies that when copying to the spool reader queue from an indexed file, only those records whose key is less than or equal to the characters specified on this parameter are to be copied. The number of characters that can be included on this parameter is from 1 through 29.
LOREC-number	This optional parameter specifies that the records whose relative record number in the file is equal to or greater than the number specified on this parameter are to be copied to the spool reader queue. The copy operation is stopped by either the last record on the file or the number on the HIREC parameter.
HIREC-number	This optional parameter specifies that the records whose relative record number in the file is equal to or less than the number specified on this parameter are to be copied to the spool reader queue. The copy operation begins with either the first record on the file or the number on the LOREC parameter.
OUTPUT-{FILE / ADD}	This optional parameter causes jobs to be copied from the spool reader queue to a file (OUTPUT-FILE); or, if OUTPUT-ADD is specified, the spool reader queue records are added to the current logical end of the file rather than at the beginning. If the OUTPUT parameter is not specified, the default operation will copy records from a file to the spool reader queue.
JOBN-{jobname / characters**}	This optional parameter specifies that when copying from the spool reader queue to a file, only those jobs are to be copied whose jobname either matches the JOBN parameter or begins with the same characters that precede the **.

Parameter	Description
COPYRDRQ Statement (continued)	
PARTITION-{1 / 2 / 3}	This optional parameter specifies that when copying from the spool reader queue to a file, only those jobs which will execute in the partition specified on the PARTITION parameter are to be copied.
REMOVE-{NO / YES / ONLY}	When copying from a reader queue to a file, this optional parameter specifies whether the job is only copied (REMOVE-NO), copied and removed from the reader queue (REMOVE-YES), or removed from the reader queue without being copied (REMOVE-ONLY).
COPYCTRL Statement	
FILE-{CONTROL / filename}	Specifies the name of the file containing the control statements. A FILE statement that defines this file must be included in the OCL when CCP is started.
DISPLAY Statement	
UNIT-code	Identifies the location of the spool file. Possible codes are those for the main data areas. If this parameter is not specified, the copy output is from the queues on the active spool file.
OUTPUT-{FILE / TERMINAL / ADD}	Directs the copy output to the file identified in the FILE parameter. Or, if the program is executing under control of CCP, the copy output can be directed to the CCP terminal that made the request (OUTPUT-TERMINAL). The OUTPUT-ADD parameter causes the same operation as the OUTPUT-FILE parameter, except the output records are added to the current logical end of file rather than at the beginning.
FILE-{DISPLAYQ / filename}	Specifies the name of the file that is to receive the copy output. A FILE statement defining this file must be included in the OCL when the spool file program is loaded.
Q-{WQ / RQ / PQ}	Specifies the queue to be displayed. WQ specifies the print queue, RQ specifies the reader queue, and PQ specifies the punch queue.
RESTORE Statement	
FILE-{RESTORE / filename}	Specifies the name of the file from which the print or punch queue records are copied. An OCL FILE statement that defines this file must be included in the OCL statements when the spool file copy program is loaded.
JOBN-{jobname / characters**}	Copies the print or punch queue records for only those jobsteps whose jobname either matches the JOBN parameter or begins with the same characters that precede the **.
STEPN-{stepname / characters**}	Copies the print or punch queue records for only those job steps whose stepname either matches the STEPN parameter or begins with the same characters that precede the **. When the STEPN parameter is specified, the JOBN parameter must also be specified.

Parameter	Description
RESTORE Statement (continued)	
{FORMSNO / CARDNO}-xxx	Copies the print queue records for only those job steps whose forms type matches the FORMSNO parameter, or copies the punch queue records for only those job steps whose card type matches the CARDNO parameter.
STOP-{STEP / FORM / CARD / NO}	Specifies whether a message is to be issued before each job step is copied (STOP-STEP), when the forms type changes (STOP-FORM), or when the card type changes (STOP-CARD); the default parameter (STOP-NO) specifies that no message is to be issued. The message, if there is one, contains either the forms type and the number of pages for the next job step to be copied or the card type for the next job step to be copied.
OUTPUT-{SPOOL / PRINT / PUNCH}	Specifies that the records copied from the file are to be either placed on the appropriate spool queue (OUTPUT-SPOOL) or copied to the print or punch device (OUTPUT-PRINT or OUTPUT-PUNCH).
UNIT-code	Identifies the location of the spool file containing the print or punch queue to which the records from the file are to be copied. If this parameter is not specified, the active spool file is assumed.
COPYQ Statement	
Q-{WQ / RQ / PQ}	Specifies the queue from which job steps are to be copied. WQ specifies the print queue, RQ specifies the reader queue, and PQ specifies the punch queue.
FROM-{code / SP}	Specifies either the main data area that contains the spool file from which the queue is to be copied (FROM-code), or that the queue is to be copied from the active spool file (FROM-SP). The FROM-SP parameter is valid only if spool is active.
TO-{code / SP}	Specifies either the main data area (TO-code) or the active spool file (TO-SP) is to receive the copied queue. If TO-code is specified and a spool file does not exist on this main data area, the spool file copy program creates one (SCP only). The TO-SP parameter is valid only if spool is active.
JOBN-{jobname / characters**}	Copies only those job steps whose jobname either matches the JOBN parameter or begins with the same characters that precede the **.
STEPN-{stepname / characters**}	Copies only those job steps whose stepname either matches the STEPN parameter or begins with the same characters that precede the **. When STEPN is specified, JOBN must also be specified.
REMOVE-{NO / YES}	Specifies whether the job step is to be removed from the queue after being copied.
PARTITION-{1 / 2 / 3}	Copies only those jobs which can be executed in the partition specified on the PARTITION parameter. If this parameter is specified, the reader queue must also be specified (Q-RQ).

Parameter	Description
COPYQ Statement (continued)	
PRIORITY-{1 / 2 / 3 / 4 / 5}	Copies only those job steps whose priority matches the PRIORITY parameter.
{FORMSNO / CARDNO}-xxx	Copies only those job steps whose forms type matches the FORMSNO parameter, or copies only those job steps whose card type matches the CARDNO parameter. If either of these parameters is specified, the copy must not be for the reader queue.
AUTHORIZE Statement	
LIST-{NO / YES}	Specifies whether a listing of authorization records is to be printed on the system printer after all changes to the authorization file have been made.
CLASSIFY Statement	
PROGRAM-programname	Specifies the name of the program that is being assigned a class number.
UNIT-code	Specifies the location of the library containing the program to be classified. Possible codes are those for the simulation areas.
CLASS-{0 / 1 / 2 / 3}	Specifies the class number to be assigned to the program.
LIBRARY-{O / R}	Specifies whether the program to be classified is in the O or the R library. If this parameter is not specified, the O library is assumed.
PACK-packname	Specifies the name of the simulation area indicated by the UNIT parameter. If this parameter is not specified, the desired pack is assumed to be on the unit specified.

This page intentionally left blank.

PARAMETER DESCRIPTIONS—COPYSP

FROM and TO Parameters

FROM specifies the device from which the spool file is copied. The TAPE (FROM-TAPE) parameter, the SP (FROM-SP) parameter, or one of the following main data area codes can be specified: D1, D2, D3, D31, D32, D33, D34, D4, D41, D42, D43, or D44. The tape unit must be specified by an OCL FILE statement with the filename $SPOOL. The copy from tape function can be used only for a tape that was previously created by the COPYSP function of the spool file copy program.

TO specifies the device to which the spool file is copied. Either TAPE (TO-TAPE) or the following main data area codes can be specified: D1, D2, D3, D31, D32, D33, D34, D4, D41, D42, D43, or D44. The tape unit must be specified by an OCL FILE statement with the filename $SPOOL. The spool file program does not support a tape-to-tape (FROM-TAPE, TO-TAPE) operation.

FROM-SP causes the active spool file to be copied and is valid only when the spool is active.

PARAMETER DESCRIPTIONS—COPYPRTQ

UNIT Parameter

This optional parameter identifies the main data area containing the spool file from which the print queue is copied. One of the following main data area codes can be specified: D1, D2, D3, D31, D32, D33, D34, D4, D41, D42, D43, or D44. If this parameter is not specified, job steps are copied from the print queue on the active spool file.

FORMSNO Parameter

This optional parameter specifies that only those job steps whose forms type matches that supplied in the FORMSNO parameter are to be copied.

JOBN Parameter

This optional parameter specifies that only those job steps whose jobname matches that supplied in the JOBN parameter are to be copied. Copying multiple jobs, whose jobnames begin with the same characters, requires the specification of those characters followed by **. From 1 through 7 characters can precede the **.

STEPN Parameter

This optional parameter specifies that only those job steps whose stepname matches that supplied in the STEPN parameter are to be copied. Copying multiple job steps, whose stepnames begin with the same characters, requires the specification of those characters followed by **. From 1 through 7 characters can precede the **. When the STEPN parameter is included, the JOBN parameter must also be included.

LENGTH Parameter

This optional parameter specifies that the data placed in the file be expanded to 134 bytes (LENGTH-FLR) or remain compressed as it exists on the spool file and placed contiguously in 134-byte records in the output file (LENGTH-VLR). If this parameter is not included, FLR is assumed.

REMOVE Parameter

This optional parameter specifies the following: the job step is to be removed from the print queue when the copy operation is complete (REMOVE-YES), the job step is to be removed from the print queue without being copied (REMOVE-ONLY), or the status of the job step is to remain unchanged on the print queue after being copied (REMOVE-NO). For job steps with multiple copies and REMOVE-YES specified, the number of copies is reduced by one after the copy is complete and the job step remains on the print queue. If REMOVE-ONLY is specified, the jobsteps will be removed regardless of the number of copies. If this parameter is not included, NO is assumed.

OUTPUT Parameter

This optional parameter specifies that the copy of the print queue either be directed to the file identified by the FILE parameter or be printed. If directed to the file, the control information (Q- and R-bytes) is included. If OUTPUT-PRINT is specified, the format of the printed output is the same as if the job had not been spooled. When OUTPUT-PRINT is specified, LENGTH-VLR is invalid. If the program is executing under CCP, OUTPUT-PRINT cannot be specified. The OUTPUT-ADD parameter specifies that the output operation is to be the same as for OUTPUT-FILE except that if the output file is a disk file, the records are added to the current logical end of the file.

FILE Parameter

This optional parameter allows you to specify the name of the file to which job steps on the print queue are to be copied. When the spool file program is loaded, you must include an OCL FILE statement that defines this file.

If the FILE parameter is not supplied, the filename PRINTQ is used by the spool file copy program. If either OUTPUT-PRINT or REMOVE-ONLY is specified the FILE parameter is ignored and the file is not used.

HEADER Parameter

This optional parameter specifies whether or not a record, which contains information about the job step to be copied, should be included immediately preceding the job step. If HEADER-YES is specified, the record is included; if HEADER-NO is specified or assumed, the record is not copied. If OUTPUT-PRINT is specified, the HEADER-YES parameter does not have any effect.

STOP Parameter

This optional parameter specifies whether a message is to be issued that indicates the forms type and the number of pages for the job step to be copied. The message can be issued before each step is copied (STOP-STEP) or before a change in forms type (STOP-FORM). The message is not issued if STOP-NO is specified or assumed. In addition to the STOP-STEP or STOP-FORM parameter, the message is issued only if OUTPUT-PRINT is specified and the spool file copy program is executing in a nonspooled partition.

PARAMETER DESCRIPTIONS—COPYPCHQ

UNIT Parameter

This optional parameter identifies the location of the spool file containing the punch queue to be copied. One of the following main data area codes can be specified: D1, D2, D3, D31, D32, D33, D34, D4, D41, D42, D43, or D44. If this parameter is not specified, job steps are copied from the punch queue on the active spool file.

CARDNO Parameter

This optional parameter specifies that only those job steps whose card type matches that supplied in the CARDNO parameter are to be copied.

JOBN Parameter

This optional parameter specifies that only those job steps whose jobname matches that supplied in the JOBN parameter are to be copied. Copying multiple jobs, whose jobnames begin with the same characters, requires the specification of those characters followed by **. From 1 through 7 characters can precede the **.

STEPN Parameter

This optional parameter specifies that only those job steps whose stepname matches that supplied in the STEPN parameter are to be copied. Copying multiple job steps, whose stepnames begin with the same characters, requires the specification of those characters followed by **. From 1 through 7 characters can precede the **. When the STEPN parameter is included, the JOBN parameter must also be included.

REMOVE Parameter

This optional parameter specifies the following: the job step is to be removed from the punch queue when the copy operation is complete (REMOVE-YES), the job step is to be removed from the punch queue without being copied (REMOVE-ONLY), or the status of the job step is to remain unchanged on the punch queue after being copied (REMOVE-NO). For job steps with multiple copies and REMOVE-YES specified, the number of copies is reduced by one after the copy is complete and the job step remains on the punch queue. If this parameter is not included, NO is assumed.

OUTPUT Parameter

This optional parameter specifies that the copy output of the punch queue either is to contain (OUTPUT-FILE) or is not to contain (OUTPUT-PUNCH) the control information (Q- and R-bytes) used by spool to punch the records. The OUTPUT-ADD parameter specifies that the output operation is to be the same as for OUTPUT-FILE except that if the output file is a disk file, the records are added to the current logical end of the file. If this parameter is not included, FILE is assumed.

FILE Parameter

This optional parameter allows you to specify the name of the file to which the job steps on the punch queue are copied. When the spool file copy program is loaded, you must include an OCL FILE statement that defines this file.

If the FILE parameter is not supplied, the filename PUNCHQ is used by the spool file copy program. If REMOVE-ONLY is specified, the FILE parameter is ignored and the file is not used.

HEADER Parameter

This optional parameter specifies whether or not a header record, which contains information about the job step is to be copied, is to be included immediately preceding the job step. If HEADER-YES is specified, the header record is included; if HEADER-NO is specified or assumed, the header record is not copied. If OUTPUT-PUNCH is specified, the HEADER-YES parameter does not have any effect.

STOP Parameter

This optional parameter specifies whether a message is to be issued that indicates the card type for the job step to be copied. The message can be issued before each step is copied (STOP-STEP) or before a change in card type (STOP-CARD). The message is not issued if STOP-NO is specified or assumed. In addition to the STOP-STEP or the STOP-CARD parameter, the message can be issued only if all of the following conditions exist: OUTPUT-PUNCH is specified, the spool file copy program is executing in a nonspooled partition, and the punch queue records being copied are for the same device as the output file (punch) device.

PARAMETER DESCRIPTIONS—COPYRDRQ

UNIT Parameter

This optional parameter specifies the location of the spool file that contains the reader queue to or from which jobs are to be copied. Possible codes are D1, D2, D3, D31, D32, D33, D34, D4, D41, D42, D43, and D44. If this parameter is not specified, jobs are copied to or from the reader queue on the active spool file.

RECL Parameter

When you are copying to the spool reader queue, this optional parameter specifies the length of records on the spool reader queue. This parameter is required if the copy is to a spool file on a system that does not have either spool active or spool reader support. If the spool reader records are to be used in conjunction with an 80-column spool reader device, then RECL-80 must be specified. Similarly, a 96-column spool reader device requires that RECL-96 be specified.

This parameter is optional when copying to a spool file on a system with spool active and the spool reader supported. A default record length is assumed based on the spool reader device. For more information about the spool reader queue record length, refer to the *Spool File Considerations and Restrictions (COPYRDRQ).*

If this parameter is specified, the OUTPUT, JOBN, PARTITION, and REMOVE parameters cannot be specified.

INPUT Parameter

This optional parameter identifies the source of the records copied to the reader queue. If INPUT-FILE is specified, the records contained in the READERQ file or the file specified by the FILE parameter are copied to the spool reader queue. If INPUT-READER is specified, records read from the system input device are copied to the spool reader queue. (The spool file copy program must be executing under control of SCP when INPUT-READER is specified.) If INPUT-TERMINAL is specified, records sent from the terminal that made the program request are copied to the spool reader queue. (The spool file copy program must be executing under control of CCP when INPUT-TERMINAL is specified.) If this parameter is not included, FILE is assumed. If this parameter is specified, the OUTPUT, JOBN, PARTITION, and REMOVE parameters cannot be specified.

FILE Parameter

This optional parameter specifies either the name of the file from which jobs are copied to the spool reader queue or the name of the file to which jobs are copied from the spool reader queue. A FILE statement that defines this file must be included in the OCL when the spool file copy program is loaded.

If this parameter is not included, the spool file copy program uses the default name READERQ.

If INPUT-READER, INPUT-TERMINAL, or REMOVE-ONLY with the OUTPUT parameter is specified, the FILE parameter is ignored and the file is not used.

KEY Parameter

This optional parameter is only for copying to the spool reader queue from an indexed file. A minimum of 1 character and a maximum of 29 characters can be included on this parameter. These characters are compared, character by character, starting with the leftmost character in the key. The compare operation ends when the last character is detected either in the key or on the parameter (whichever is first). The record is not copied unless an equal condition exists.

If a comma, hyphen, blank, or an apostrophe (single quote) is included in the specified characters, the character string must be enclosed by apostrophes. An apostrophe can be included in the character string only if it is specified in two consecutive positions.

If the KEY parameter is specified, the LOKEY, HIKEY, LOREC, HIREC, OUTPUT, JOBN, PARTITION, and REMOVE parameters cannot be specified.

LOKEY Parameter

This optional parameter is only for copying to the spool reader queue from an indexed file. The characters included on this parameter determine the first record to be copied from the file. Only those records that have a key equal to or greater than the characters on this parameter are copied. A minimum of 1 character and a maximum of 29 characters can be included on this parameter.

If a comma, hyphen, blank, or an apostrophe (single quote) is included in the specified characters, the character string must be enclosed by apostrophes. An apostrophe can be included in the character string only if it is specified in two consecutive positions.

If the characters specified on this parameter are not the same length as the keys for the file records, the characters specified are truncated or padded on the right with blanks (hex 40) as needed to match the key length.

If this parameter is specified, the KEY, LOREC, HIREC, OUTPUT, JOBN, PARTITION, and REMOVE parameters cannot be specified.

If this parameter is not specified and the HIKEY parameter is specified, the low key default value is blanks (hex 40).

HIKEY Parameter

This optional parameter is only for copying to the spool reader queue from an indexed file. This parameter specifies the highest key for which a record can be copied. Only those records are copied with a key equal to or less than the character(s) on this parameter.

If this parameter is specified, the KEY, LOREC, HIREC, OUTPUT, JOBN, PARTITION, and REMOVE parameters cannot be specified.

If this parameter is not specified and LOKEY is specified, the default value for this parameter is all nines (hex F9).

A minimum of 1 character and a maximum of 29 characters can be included on this parameter.

If a comma, hyphen, blank, or an apostrophe (single quote) is included in the specified characters, the character string must be enclosed by apostrophes. An apostrophe can be included in the character string only if it is specified in two consecutive positions.

If the characters specified on this parameter are not the same length as the keys for the file records, the characters specified are truncated or padded on the right with nines (hex F9) as needed to match the key length.

LOREC Parameter

When copying to the spool reader queue from a file, this optional parameter specifies the relative record number from which the copy operation is to begin. Relative record numbers identify a record's location with respect to other records in the file. The relative record number of the first record is 1, the number of the second record is 2, and so on. A positive decimal number that consists of a minimum of 1 digit and a maximum of 15 digits can be specified.

If this parameter is specified, the KEY, LOKEY, HIKEY, OUTPUT, JOBN, PARTITION, and REMOVE parameters cannot be specified.

If this parameter is not specified and the HIREC parameter is specified, the relative record number default is 1.

HIREC Parameter

When copying to the spool reader queue from a file, this optional parameter specifies the relative record number which ends the copy operation (this record is copied). Relative record numbers identify a record's location with respect to other records in the file. The relative record number of the first record is 1, the number of the second record is 2, and so on. A positive decimal number that consists of a minimum of 1 digit and a maximum of 15 digits can be specified.

If this parameter is specified, the KEY, LOKEY, HIKEY, OUTPUT, JOBN, PARTITION, and REMOVE parameters cannot be specified.

If this parameter is not specified and the LOREC parameter is specified, the relative record number default is the highest record number in the file.

OUTPUT Parameter

This optional parameter causes jobs to be copied from the spool reader queue to a file (OUTPUT-FILE). If OUTPUT-ADD is specified and the output file is a disk file, the spool reader queue records are added to the current logical end of the file, rather than at the beginning. If the OUTPUT parameter is not specified, the default operation will copy records from a file to the spool reader queue. If this parameter is specified, the INPUT, RECL, KEY, LOKEY, HIKEY, LOREC, and HIREC parameters cannot be specified.

JOBN Parameter

When copying from the spool reader queue to a file, this optional parameter specifies that only those jobs whose jobname matches the JOBN parameter are to be copied. Copying multiple jobs, whose jobnames begin with the same characters, requires the specification of those characters followed by **. From 1 through 7 characters can precede the **. If this parameter is specified, the OUTPUT parameter must also be specified; the INPUT, RECL, KEY, LOKEY, HIKEY, LOREC, and HIREC parameters cannot be specified.

PARTITION Parameter

When copying from the spool reader queue to a file, this optional parameter specifies that only those jobs which can be executed in the partition specified on the PARTITION parameter are to be copied. If this parameter is specified, the OUTPUT parameter must also be specified; the INPUT, RECL, KEY, LOKEY, HIKEY, LOREC, and HIREC parameters cannot be specified.

REMOVE Parameter

When copying from the spool reader queue to a file, this optional parameter specifies that: the job is to be removed from the reader queue when the copy operation is complete (REMOVE-YES), the job is to be removed from the reader queue without being copied (REMOVE-ONLY), or the status of the job is to remain unchanged on the reader queue after being copied (REMOVE-NO). If this parameter is not specified, REMOVE-NO is assumed. If this parameter is specified, the OUTPUT parameter must also be specified; the INPUT, RECL, KEY, LOKEY, HIKEY, LOREC, and HIREC parameters cannot be specified.

PARAMETER DESCRIPTIONS—COPYCTRL

FILE Parameter

This optional parameter allows you to specify the name of the file that contains the control statements. The default name CONTROL is used if this parameter is not included.

A FILE statement defining this file must be included in the OCL statements when CCP is started.

PARAMETER DESCRIPTIONS—DISPLAY

UNIT Parameter

This optional parameter specifies the location of the spool file to be displayed. Possible codes are: D1, D2, D3, D31, D32, D33, D34, D4, D41, D42, D43, and D44. If this parameter is not specified, the status of the active spool file is displayed.

OUTPUT Parameter

This parameter is optional. If OUTPUT-FILE is specified, the copy output is directed to the file identified in the FILE parameter or to the file named DISPLAYQ.

The OUTPUT-ADD parameter specifies that the output operation is to be the same as for OUTPUT-FILE except that if the output file is a disk file, the records are added to the current logical end of the file.

If the spool file copy program is executing under CCP and OUTPUT-TERMINAL is specified, the copy output is directed to the CCP terminal that made the program request.

If this parameter is not included, OUTPUT-FILE is assumed when the program is executing under SCP, and OUTPUT-TERMINAL is assumed when the program is executing under CCP and has a terminal. The OUTPUT-TERMINAL parameter is not valid when the spool file copy program is executing under SCP or under CCP and does not have a terminal.

FILE Parameter

This optional parameter allows you to specify the name of the file to which the display output is copied. If the copy output is to a file (OUTPUT-FILE) and this parameter is not included, the filename DISPLAYQ is assumed.

If the display output is copied to a file, a FILE statement defining this file must be included in the OCL statements when the spool file copy program is loaded.

If OUTPUT-TERMINAL is specified, the FILE parameter is ignored.

Q Parameter

This optional parameter specifies which queue is to be displayed. If Q-RQ is specified, the reader queue is displayed. If Q-WQ is specified, the print queue is displayed. If Q-PQ is specified, the punch queue is displayed.

If OUTPUT-FILE is specified and this parameter is not included, all queues are copied. The status of the reader is copied first, followed by the print queue, then the punch queue.

If OUTPUT-TERMINAL is specified and this parameter is not included, the print queue is displayed.

PARAMETER DESCRIPTIONS—RESTORE

FILE Parameter

This optional parameter specifies the name of the file that contains the print or punch queue records to be copied. If this parameter is not included, RESTORE is assumed to be the name of the file. A FILE statement that defines this file as being either a disk file or a tape file must be included in the OCL statements when the spool file copy program is loaded.

JOBN Parameter

This optional parameter specifies that the print or punch queue records are to be copied for only those job steps whose jobname matches the name supplied on the JOBN parameter. Copying multiple jobs, whose jobnames begin with the same characters, requires the specification of those characters followed by **. From 1 through 7 characters can precede the **. When this parameter is specified, a job step in the file must have a header record in order to be copied.

STEPN Parameter

This optional parameter specifies that the print or punch queue records are to be copied for only those job steps whose stepname matches the name supplied on the STEPN parameter. Copying multiple job steps, whose stepnames begin with the same characters, requires the specification of those characters followed by **. From 1 through 7 characters can precede the **. If the STEPN parameter is included, the JOBN parameter must also be included. When this parameter is specified, a job step in the file must have a header record in order to be copied.

FORMSNO Parameter

This optional parameter specifies that the print queue records are to be copied for only those job steps whose forms type matches this parameter.

When this parameter is specified, a job step in the file must have a header record in order to be copied.

The FORMSNO parameter and the CARDNO parameter are mutually exclusive.

CARDNO Parameter

This optional parameter specifies that the punch queue records are to be copied for only those job steps whose card type matches this parameter.

When this parameter is specified, a job step in the file must have a header record in order to be copied.

The CARDNO parameter and the FORMSNO parameter are mutually exclusive.

STOP Parameter

This optional parameter specifies whether a message should be issued for the job step to be copied. The message indicates the forms type and the number of pages for the print records or the card type for the punch records. If STOP-STEP is specified, the message is issued before each step is copied. If STOP-FORM or STOP-CARD is specified, the message is issued before each step is copied in which the forms type or the card type is different than in the preceding step. If STOP-NO is specified or assumed, the message is not issued. In addition, the message can only be issued if the spool file copy program is executing in a nonspooled partition, the job step to be copied has a header record, and the job step is being copied to the print or punch device.

OUTPUT Parameter

This optional parameter specifies whether the print or punch records copied from the file are to be placed on the spool print or spool punch queue (OUTPUT-SPOOL) or to be printed or punched on the device for which they were originally intercepted by spool (OUTPUT-PRINT or OUTPUT-PUNCH). If this parameter is not specified, OUTPUT-SPOOL is assumed. In order to be copied from the file to the spool queue, a job step must have a header record.

UNIT Parameter

This optional parameter specifies the location of the spool file that contains the print or punch queue to which the records are to be copied from the file. Possible codes are D1, D2, D3, D31, D32, D33, D34, D4, D41, D42, D43, and D44. If this parameter is not specified, the print or punch queue on the active spool file is assumed.

PARAMETER DESCRIPTIONS—COPYQ

Q Parameter

This parameter specifies the spool queue for which job steps are to be copied. If Q-WQ is specified, the print queue job steps are copied. If Q-RQ is specified, the reader queue jobs are copied. If Q-PQ is specified, punch queue job steps are copied.

FROM Parameter

This parameter specifies the location of the main data area that contains the spool file from which the queue is to be copied. Either SP (FROM-SP) or one of the following main data area codes must be specified: D1, D2, D3, D31, D32, D33, D34, D4, D41, D42, D43, or D44. If SP is specified, the queue is copied from the active spool file.

TO Parameter

This parameter specifies the location of the main data area that is to receive the copy of the queue. If a spool file does not exist on this main data area, the spool file copy program creates one (SCP only).

Either SP (TO-SP) or one of the following main data area codes must be specified: D1, D2, D3, D31, D32, D33, D34, D4, D41, D42, D43, or D44. If SP is specified, the queue is copied to the active spool file.

JOBN Parameter

This optional parameter specifies that only those job steps whose jobname matches that name supplied on the JOBN parameter are to be copied. Copying multiple jobs, whose jobnames begin with the same characters, requires the specification of those characters followed by **. From 1 through 7 characters can precede the **.

STEPN Parameter

This optional parameter specifies that only those job steps whose stepname matches that name supplied on the STEPN parameter are to be copied. Copying multiple job steps, whose stepnames begin with the same characters, requires the specification of those characters followed by **. From 1 through 7 characters can precede the **. This parameter cannot be included when copying the reader queue. When the STEPN parameter is included, the JOBN parameter must also be included.

REMOVE Parameter

This optional parameter specifies that either the copied job step be removed from the queue (REMOVE-YES) or the status of the job step remain unchanged on the queue (REMOVE-NO). For job steps with multiple copies and REMOVE-YES specified, the number of copies is reduced by one after the copy is complete; the job step remains on the queue. If this parameter is not included, REMOVE-NO is assumed.

PARTITION Parameter

This optional parameter specifies that only those jobs are to be copied which can be executed in the partition specified on the PARTITION parameter. Possible partition parameters are 1, 2, and 3.

This parameter can only be included if the copy is for the reader queue.

PRIORITY Parameter

This optional parameter specifies that only those job steps whose priority matches that supplied in the PRIORITY parameter are to be copied. Possible priority parameters are 1, 2, 3, 4, and 5.

FORMSNO and CARDNO Parameter

This optional parameter specifies that the job step(s) is to be copied only when the forms type and the FORMSNO parameter match (print) or when the card type and the CARDNO parameter match (punch). The FORMSNO and CARDNO parameters are mutually exclusive.

If this parameter is included, the copy must not be for the reader queue.

PARAMETER DESCRIPTIONS-AUTHORIZE

LIST Parameter

This optional parameter (LIST-YES or LIST-NO) specifies whether the contents of the AUTHORIZ file is to be printed. When LIST-YES is specified, the contents of the AUTHORIZ file are printed on the system printer after all changes to the AUTHORIZ file have been made. When LIST-NO is specified, the contents of the AUTHORIZ file are not printed. If this parameter is not specified, LIST-NO is assumed.

PARAMETER DESCRIPTIONS-CLASSIFY

PROGRAM Parameter

This parameter (PROGRAM-program name) specifies the name of the program that is to be assigned a class number.

UNIT Parameter

This parameter (UNIT-code) specifies the location of the simulation area that contains the library containing the program to be assigned a class number. Possible codes are F1, R1, F2, or R2.

CLASS Parameter

This parameter (CLASS-0, 1, 2, or 3) specifies the class number to be assigned to the program identified by the PROGRAM parameter.

LIBRARY Parameter

This optional parameter (LIBRARY-O or LIBRARY-R) specifies whether the program to be assigned a class number is in the O (object) library or the R (routine) library. If this parameter is not specified, the O library is assumed.

PACK Parameter

This optional parameter (PACK-packname) verifies that the correct library is accessed when a program is assigned a class number. (The PACK parameter is compared with the name of the simulation area identified by the UNIT parameter.) If the PACK parameter is not specified, verification is not done.

SPOOL FILE CONSIDERATIONS AND RESTRICTIONS

FILE Requirements

Only those FILE statements that are needed to execute the desired functions of $QCOPY need to be included in the OCL.

The spool file copy program functions that require files also require data management subroutines to be included to access those files. The function requested is compared with the specified file to determine which data management subroutines are to be included. These subroutines are then linked together as required immediately before the function is executed. The required data management subroutines must be located in the R-library of either the system pack or the program pack.

Figure 4-94 shows the files and the data management subroutines required for specific functions of the spool file copy program.

Partition Size Requirements

Figure 4-94 shows the minimum partition sizes required to execute specific functions of $QCOPY. However, the performance of $QCOPY might be improved when this program is executed in a partition with a size larger than minimum requirements (16K, for example). The larger partition size allows $QCOPY to use a larger I/O buffer area because the $QCOPY functions that involve copying to or from a spool file use the available space in the partition for I/O buffers, both for the spool file and for the data management (if disk files are used). Under CCP, this capability requires the specification of a larger value on the TASKSIZE parameter of the PROGRAM statement in the assignment set.

Copy the Entire Spool File (COPYSP)

Tape-to-tape copy is not supported.

The FROM and TO parameters cannot be the same.

To minimize the time required to execute the spool file copy program, only spool file track groups containing data are copied.

The spool file, when copied to disk, can be used by spool after an IPL is performed.

Any job steps that are incomplete when the copy is started will be incomplete on the copy of the spool file. To prevent copying incomplete job steps, the COPYSP function from an active spool file should not be started until all spooling functions in the supervisor are complete and all spooled partitions are at end of job.

If the copy from the active spool file is started before all spooling functions in the supervisor are complete and before all spooled partitions are at end of job, then the spool file copy program can cause the spooling functions to wait for the copy to complete.

When a spool file is copied either to or from a tape file, a record length of 1024 is used. A default block length of 1024 is used unless otherwise specified on the tape OCL FILE statement.

The spool file copy program supports copying an inactive spool file to a multivolume tape file but does not support copying an active spool file to a multivolume tape file. Therefore, when the active spool file is being copied to tape and the end-of-reel marker is detected, a message is issued and the copy function is discontinued. You can overcome this situation by copying the active spool file to disk first and then copying the new inactive spool file from disk to a multivolume tape file.

The COPYSP control statement is valid only when the spool file copy program is executed under the system control program.

Copy Selected Job Steps from the Print Queue (COPYPRTQ)

A job step on the print queue cannot be copied if any of the following conditions exist:

- The job step is currently executing.
- The job step is being printed by spool or being copied by the spool file copy program in another partition.
- A REUSE command was specified for the job step.
- The QCOPY-NO parameter had been specified on the PRINTER OCL statement for that job step.

When COPYPRTQ is specified without the FORMSNO, JOBN, and STEPN parameters, all available job steps on the print queue (including those that are currently held) are copied.

When a job step is being copied from the print queue, that job step appears on a display of the queue with an indication that it is being copied by $QCOPY. However, system commands that normally pertain to that job step do not affect it.

Fixed-length records (FLR) written to the PRINTQ file contain printer IOB Q- and R-bytes as the first 2 bytes. (The Q- and R-bytes contain data that is used by spool to print the record.) An end-of-step record, consisting of 134 bytes with hex FFFF in the first 2 bytes, is written at the end of each step for each step that is completely copied.

Each variable-length record (VLR) written to the PRINTQ file contains a length byte (which defines the total length of the record including the length byte) followed by the printer IOB Q- and R-bytes and the print record with the trailing blanks removed. Multiple VLR records are blocked into a 134-byte record and then written to the file. An end-of-step record consisting of hex 03FFFF is appended to the VLR print records in the buffer at end of step for each step that is completely copied. The print records for the next step start at the beginning of the next 134-byte record.

A form length change record, consisting of hex 04E100nn for VLR or hex E100nn plus an additional 131 bytes for FLR, appears anywhere among the print records to indicate a change in the forms length. The nn indicates the new forms length in hex.

This page intentionally left blank.

Function	Default File Name	File Device	File Access Method	File Record Length[2]	Minimum Partition Size	List of Required Data Management (See Part 4)
COPYSP						
FROM-TAPE	$SPOOL	Tape	Input	1024	10K Plus 2 Times the Block Length[1]	1
TO-TAPE		Tape	Output	1024	10K	2
COPYPRTQ						
OUTPUT-FILE or OUTPUT-ADD	PRINTQ	Unit Record	Output	134	10K	3
REMOVE-YES or REMOVE-NO		Simulation Area	Consecutive Add or Consecutive Output	134	10K	4
		Main Data Area	Consecutive Add or Consecutive Output	134	10K	5
		Tape	Output	134	10K Plus 2 Times the Block Length[1]	3
COPYPCHQ						
OUTPUT-FILE or OUTPUT-ADD	PUNCHQ	Unit Record	Output	MFCU-98 MFCM-83 1442-82	10K	3
REMOVE-YES or REMOVE-NO		Simulation Area	Consecutive Add or Consecutive Output	MFCU-98 MFCM-83 1442-82	10K	4
		Main Data Area	Consecutive Add or Consecutive Output	MFCU-98 MFCM-83 1442-82	10K	5
		Tape	Output	MFCU-98 MFCM-83 1442-82	10K Plus 2 Times the Block Length[1]	3
OUTPUT-PUNCH		Unit Record	Output	MFCU-96 MFCM-80 1442-80	10K	3
REMOVE-YES or REMOVE-NO		Simulation Area	Consecutive Add or Consecutive Output	MFCU-96 MFCM-80 1442-80	10K	4
		Main Data Area	Consecutive Add or Consecutive Output	MFCU-96 MFCM-80 1442-80	10K	5

[1] Round the partition size up to the next 2K boundary.
[2] The file record length for the COPYPCHQ function depends on which punch device was being spooled.

Figure 4-94 (Part 1 of 4). Spool File Copy Program Files and Subroutines

Function	Default File Name	File Device	File Access Method	File Record Length[2]	Minimum Partition Size	List of Required Data Management (See Part 4)
COPYPCHQ (continued)		Tape	Output	MFCU-96 MFCM-80 1442-80	10K Plus 2 Times the Block Length[1]	3
COPYRDRQ						
OUTPUT-FILE or OUTPUT-ADD	READERQ	Unit Record	Output	96	10K	3
		Simulation Area	Consecutive Add or Consecutive Output	96	10K	4
REMOVE-YES or REMOVE-NO		Main Data Area	Consecutive Add or Consecutive Output	96	10K	5
		Tape	Output	96	10K Plus 2 Times the Block Length[1]	3
INPUT-FILE		Unit Record	Input	1–128	10K	6
		Simulation Area	Consecutive Input	1–128	10K	7
		Main Data Area	Consecutive Input	1–128	10K	8
		Tape	Input	1–128	10K Plus 2 Times the Block Length[1]	6
LOREC or HIREC		Simulation Area	Direct Input	1–128	10K	9
		Main Data Area	Direct Input	1–128	10K	10
KEY, LOKEY, or HIKEY		Main Data Area	Indexed Sequential Input Within Limits	1–128	12K	11
(Indexed file, keys not specified)		Main Data Area	Indexed Sequential Input	1–128	12K	12
DISPLAY						
OUTPUT-FILE or OUTPUT-ADD	DISPLAYQ	Unit Record	Output	40	10K	3
		Simulation Area	Consecutive Add or Consecutive Output	40	10K	4
		Main Data Area	Consecutive Add or Consecutive Output	40	10K	5
		Tape	Output	40	10K Plus 2 Times the Block Length[1]	3

[1] Round the partition size up to the next 2K boundary.
[2] The file record length for the COPYPCHQ function depends on which punch device was being spooled.

Figure 4-94 (Part 2 of 4). Spool File Copy Program Files and Subroutines

Function	Default File Name	File Device	File Access Method	File Record Length[2]	Minimum Partition Size	List of Required Data Management (See Part 4)
RESTORE	RESTORE	Simulation Area	Consecutive Input	Records for: 1403-134 MFCU-98 MFCM-83 1442-82	12K	7
		Main Data Area	Consecutive Input	Records for: 1403-134 MFCU-98 MFCM-83 1442-82	12K	8
		Tape	Input	Records for: 1403-134 MFCU-98 MFCM-83 1442-82	12K Plus 2 Times the Block Length[1]	6
COPYCTRL	CONTROL	Simulation Area	Direct Input	1—128	10K	N/A
		Main Data Area	Direct Input	1—128	10K	N/A
AUTHORIZE	AUTHORIZ	Main Data Area	Direct Input Direct Update	256	10K	N/A

[1] Round the partition size up to the next 2K boundary.
[2] The file record length for the COPYPCHQ function depends on which punch device was being spooled.

Figure 4-94 (Part 3 of 4). Spool File Copy Program Files and Subroutines

Required Data Management (R-Modules) for $QCOPY Functions Using Files

1	$$CSIT	$$TSMO	$$TSBS	$$TSCR						
2	$$CSOT	$$TSMO	$$TSSB	$$TSCR						
3	$$ARFF (if 1442 is used) $$LPRT (if 1403 is used)			$$MFPP (if 5424 is used) $$LPMP (if 3284 or 3287 is used)			$$MMPP (if 2560 is used) $$CPOP (if 3741 is used)			
4	$$CSOP	$$SRBR	$$SRUA	$$SRDF	$$SRTC	$$SRDI	$$SRMO	$$SRSB	$$SRBP	
5	$$CFOP	$$SFBR	$$SFUA	$$SFDF	$$SRTC	$$SFPD	$$SRMO	$$SFSB	$$SFBP	
6	$$ARFF (if 1442 is used) $$ARRD (if 2501 is used)			$$MFRD (if 5424 is used) $$CPIP (if 3741 is used)			$$MMRD (if 2560 is used)			
7	$$CSIP	$$SRBR	$$SRUA	$$SRTC	$$SRMO	$$SRSB	$$SRDI	$$SRBP		
8	$$CFIP	$$SFBR	$$SFUA	$$SRTC	$$SFMO	$$SFSB	$$SFPD	$$SFBP		
9	$$DAID	$$SRCB	$$SRDA	$$SRDI	$$SRRC	$$SRRI	$$SRTC			
10	$$DFID	$$SFPD	$$SFRC	$$SFRI	$$SFCB	$$SFDA	$$SRTC			
11	$$IHIL $$SFSI	$$SFMO	$$SFPD	$$SFIC	$$SFRC	$$SFRI	$$SFLM	$$SRTC	$$SFBP	$$SFSC
12	$$IHIP	$$SFMO	$$SFPD	$$SFIC	$$SFRC	$$SFRI	$$SRTC	$$SFBP		

Figure 4-94 (Part 4 of 4). Spool File Copy Program Files and Subroutines

This page intentionally left blank.

If HEADER-YES is specified, a 134-byte header record is written at the beginning of the output for each step. For FLR (Fixed Length Record), the first two bytes contain hex FFFE. For VLR (Variable Length Record), the first three bytes contain hex 86FFFE. The following chart gives the format of the header records following the initial 2- or 3-bytes as previously indicated:

Start Offset Hex	End Offset Hex	Length (Dec)	Description
00	07	8	Job name
08	0F	8	Step name
10	10	1	Status byte 1 X'80' Job step held on print queue X'04' Forms alignment required X'02' Halt on unprintable characters
11	11	1	Status byte 2 X'20' job step kept on print queue
12	12	1	Priority on print queue (hex)
13	13	1	Reserved
14	14	1	Forms length (hex)
15	15	1	Number of copies (hex)
16	18	3	Forms type
19	1A	2	Number of pages (hex)
1B	82(83)	104(105)	Reserved

The following chart gives the Q- and R-bytes and their meanings for the records written to the PRINTQ file:

Q-Byte	R-Byte	Description
1110 0000	0000 0000	SPACE 0
	0000 0001	SPACE 1
	0000 0010	SPACE 2
	0000 0011	SPACE 3
		An R-byte value of greater than 3 is not permitted and results in a space 0.
1110 0001	0000 0000	Forms length change[1]
1110 0010	0000 0000	Print followed by space 0
1110 0010	0000 0001	Print followed by space 1
1110 0010	0000 0010	Print followed by space 2
1110 0010	0000 0011	Print followed by space 3
		An R-byte greater than 3 is not permitted and results in a space 0.
1110 0100	0000 0001	Skip to line 1
1110 0100	0000 0010	Skip to line 2
. .	. .	.
. .	. .	.
. .	. .	.
1110 0100	0110 1111	Skip to line 111
1110 0100	0111 0000	Skip to line 112
1110 0110	0000 0001	Print followed by skip to line 1
1110 0110	0000 0010	Print followed by skip to line 2
. .	. .	.
. .	. .	.
. .	. .	.
1110 0110	0110 1111	Print followed by skip to line 111
1110 0110	0111 0000	Print followed by skip to line 112
1111 1111	1111 1110	Header record[1]
1111 1111	1111 1111	End of step record[1]

[1] Indicates Q- and R-bytes for records generated by spool and $QCOPY. They are not valid printer Q- and R-bytes.

For more information about the Q- and R-bytes, refer to the *IBM System/3 Models 8, 10, 12, 15 Components Reference Manual,* GA21-9236.

When the system is executing the spool file copy program in a spooled partition and OUTPUT-PRINT is specified (or OUTPUT-FILE with the PRINTQ file as the printer), those job steps printed by $QCOPY are intercepted by spool and placed on the queue as one job step under the job and step name used when the spool file copy program was loaded.

If the JOBN, or JOBN and STEPN parameters are specified with the FORMSNO parameter, the job steps specified are copied only if their forms type matches that given in the FORMSNO parameter.

If the output file is a disk file, it can be shared only after it is closed.

If the output file is a tape file, $QCOPY assumes a default block length of 1340 (10-records per block) unless a block length is specified on the PRINTQ OCL FILE statement. The minimum partition size when using tape is 10K bytes plus twice the tape block length rounded upward to the nearest 2K. Consider the following example:

Tape block length in bytes	1340
Multiply by 2	X2
	2680
Add base function size (10K) . .	+10240
Total storage required	12920
Round upward to next 2K	14336 (14K)

The required partition size for this example is 14K.

If the copy function is OUTPUT-FILE, punch checks can occur if the output file is a punch device, or unprintable characters can occur if the output file is a printer. These conditions can occur because the copy records contain the Q- and R-bytes as the first 2 characters. If the copy function is OUTPUT-PRINT, the spool file program supports unprintable characters in the same manner as the spool print writer. The parameter OUTPUT-PRINT is invalid when the program is executing under CCP.

When REMOVE-YES is specified, job steps are not removed from the queue until all job steps have been successfully copied. However, job steps are removed before the next control statement is read. Therefore, job steps being copied are not lost if the copy function does not terminate normally. But job steps will be lost if $QCOPY is canceled after the job steps have been removed.

When REMOVE-ONLY is specified, job steps are removed from the queue without being copied. The PRINTQ file is not used for this function.

Comments are not allowed on the COPYPRTQ statement unless at least one parameter is included.

Note: If a job step is to be added to a file in which the last job step does not have an end-of-step record (due to not being completely copied to the file from the spool queue), the job step being added must have a header record in order to be recognized from the preceding incomplete job step in the file. A job step should not be added to a file in which the last job step has variable-length print records and does not have an end-of-step record; results will be unpredictable if a RESTORE function is attempted with that file.

Copy Selected Job Steps from the Punch Queue (COPYPCHQ)

A job step on the punch queue cannot be copied if any of the following conditions exist:

- The job step is currently executing.
- The job step is being punched by spool or being copied by the spool file program in another partition.
- A REUSE command was specified for the job step.
- The QCOPY-NO parameter was specified on the PUNCH OCL statement for that job step.

When COPYPCHQ is specified without the CARDNO, JOBN, and STEPN parameters, all available job steps on the punch queue (including those that are currently held) are copied.

When a job step is being copied from the punch queue, that job step appears on a display of the queue with an indication that it is being copied by $QCOPY. However, system commands that normally pertain to that job step do not affect it.

When you specify the OUTPUT-FILE parameter, the copy output contains control bytes. These control bytes, called punch IOB Q-, R-, and H-bytes, are used by the system to punch the record. They are contained in the first (leftmost) 2 or 3 bytes of the record. For more information about the Q-, R-, and H-bytes, refer to the *IBM System/3 Models 8, 10, 12, and 15 Components Reference Manual,* GA21-9236.

Because the copy output can contain the control bytes, the record length depends on whether you specify the OUTPUT-FILE parameter or the OUTPUT-PUNCH parameter. The following chart shows the record length for the possible punch devices and the OUTPUT parameters.

	Parameters	
Punch device	**OUTPUT-PUNCH**	**OUTPUT-FILE**
5424 MFCU	96	98[1]
2560 MFCM	80	83[2]
1442 Card Read Punch	80	82[1]

[1] The first 2 bytes of the record are the punch IOB Q- and R-bytes.
[2] The first 3 bytes of the record are the punch IOB Q-, R-, and H-bytes.

If you specify the OUTPUT-FILE parameter and the copy output is directed to a punch device, the rightmost 2 or 3 bytes of the record are truncated if the record length exceeds the device capacity.

If you specify the OUTPUT-FILE parameter, an end-of-step record is written at the end of each job step that is completely copied. The first 2 bytes of this record consist of hex FFFF. Additionally, if you specify HEADER-YES, a header record that contains information about that job step is written to the file at the beginning of each copied job step. The length of the header record is the same as the record length of the file. The first two bytes of the header record consist of hex FFFE. The following chart gives the format of the header record following those initial two bytes:

Start Offset (Hex)	Ending Offset (Hex)	Length (Dec)	Description
00	07	8	Job name
08	0F	8	Step name
10	10	1	Status byte 1
			X'80' Job step held on punch queue
11	11	1	Status byte 2
			X'20' Job step kept on punch queue
12	12	1	Priority on punch queue (hex)
13	14	2	Reserved
15	15	1	Number of copies (hex)
16	18	3	Card type
19	1A	2	Number of cards (hex)
1B	End of Record		Reserved

The following chart gives the Q- and R-bytes and their meanings for the records written to the PUNCHQ file:

	Q-Byte	R-Byte	Description
1442	0101 0000		Feed
	0101 0010		Punch and Feed
		xxxx x000	Select stacker 1
		xxxx x001	Select stacker 1
5424 (MFCU) 2560 (MFCM)	1111 0000		Feed (primary)
	1111 0010		Punch and feed (primary)
	1111 1000		Feed (secondary)
	1111 1010		Punch and feed (secondary)
5424 (MFCU)		xxxx x000	No stacker selection
		xxxx x100	Select stacker 4
		xxxx x101	Select stacker 1
		xxxx x110	Select stacker 2
		xxxx x111	Select stacker 3
2560 (MFCM)		xxxx x000	Stacker select default (1—primary, 5—secondary)
		xxxx x001	Select stacker 1
		xxxx x010	Select stacker 2
		xxxx x011	Select stacker 3
		xxxx x100	Select stacker 4
		xxxx x101	Select stacker 5
1442 5424 2560	1111 1111	1111 1110	Header record[1]
	1111 1111	1111 1111	End of step record[1]

[1] Indicates Q- and R-bytes for records generated by $QCOPY and are not valid Q- and R-bytes.

For more information about Q-, R-, and H-bytes, refer to the *IBM System/3 Models 8, 10, 12, and 15 Components Reference Manual*, GA21-9236.

When the system is executing the spool file copy program in a partition with spool active, and the PUNCHQ file is the spool punch, then those job steps punched are intercepted by spool and placed on the punch queue as one job step under the job and step name used when the program was loaded.

If the output file is a disk file, it can be shared only after it is closed.

If the output file is a tape file, $QCOPY assumes a default block length of ten times the record length unless a block length is specified on the PUNCHQ OCL FILE statement. The minimum partition size when using tape is 10K bytes plus twice the tape block length rounded upward to the nearest 2K.

If the copy function is OUTPUT-FILE, punch checks can occur if the output file is a punch device or unprintable characters can occur if the output file is a print device. These conditions can occur because the copy records contain the Q- and R-bytes as the first 2 characters. Punch checks can also occur if OUTPUT-PUNCH is specified and the punch data is for a different type of punch device (for example, 1442 Card Read Punch spooled records copied to a 5424 MFCU).

When REMOVE-YES is specified, job steps are not removed from the queue until all job steps have been successfully copied. However, job steps are removed before the next control statement is read. Therefore, job steps being copied are not lost if the copy function does not terminate normally. But job steps will be lost if $QCOPY is canceled after the job steps have been removed.

When REMOVE-ONLY is specified, job steps are removed from the queue without being copied. The PUNCHQ file is not used for this function.

Comments are not allowed on the COPYPCHQ statement unless at least one parameter is included.

Note: If a job step is to be added to a file in which the last job step does not have an end-of-step record (due to not being completely copied to the file from the spool queue), the job step being added must have a header record in order to be recognized from the preceding incomplete job step in the file.

Copy Jobs to or from the Reader Queue (COPYRDRQ)

When a job is copied to the reader queue by $QCOPY executing under CCP, the name of the terminal from which $QCOPY was requested appears next to the jobname on a display of the reader queue. Additionally, the priority of jobs placed on the active spool file reader queue under CCP can be limited either by the HIPTY OCC (operator control command) or by an option of the configuration record program ($CNFIG).

When a job is copied to the reader queue, the RECL parameter is required if spool is not active or the spool reader is not supported. If the RECL parameter is specified in other than these conditions, it will only be used if the spool reader for the system is the 3741 or if the copy is to an inactive spool file.

If the spool reader is the 3741 and the RECL parameter is not specified, the input record length is the length of the source records as determined by the INPUT parameter:

INPUT Parameter	Input Record Length
TERMINAL	80
READER	Length of system input device records
FILE	Length of file records

The following chart shows how to determine the input record length. For example, if the spool reader is not the 3741 and either the copy is for the active spool file or the RECL parameter is not specified, then the input record length will be the length of the spool reader records.

Record Length Choices	Conditions							
			Spool reader 3741		Spool reader not 3741			
					RECL given		RECL not given	
	Spool not active	Spool reader not supported	RECL given	RECL not given	Active file	Inactive file	Active file	Inactive file
RECL given on control statement	X	X	X			X		
Length of input records				X				
Length of spool reader records					X		X	X

If records are being copied from a file, the record length of that file can be any value from 1 through 128.

Both the block length and the record length parameters must be specified on a tape OCL FILE statement if one of the following conditions exist: (1) the tape does not have a standard label, or (2) the REEL-BLP parameter is specified.

The minimum partition size when using tape is 10K bytes plus twice the tape block length rounded upward to the nearest 2K.

Disk files used to copy records to the reader queue can be consecutive, direct, or indexed. The following chart shows the access method used based on the file organization and parameter specifications.

Parameter Specifications	File Organization		
	Consecutive	Direct	Indexed
LOREC or HIREC	Direct input	Direct input	Invalid
KEY, LOKEY, or HIKEY	Invalid	Invalid	Indexed sequential input within limits[1]
Not LOREC, HIREC, KEY, LOKEY, or HIKEY	Consecutive input	Consecutive input	Indexed sequential input[1]

[1] Requires a 12K partition.

The copy function is terminated when two consecutive /. records are read or when end of data (EOD) is detected from the READERQ file and the file is disk or tape. If the job being copied to the spool reader queue is not completely copied when the copy reader queue function terminates, that job is taken off the queue.

When the output function of COPYRDRQ is specified, the jobstream records are copied from the spool reader queue and placed in a file with a record length of 96. If the JOBN and PARTITION parameters are not specified, all available jobs on the reader queue (including those jobs that are currently held) are copied.

A job cannot be copied from the reader queue if any of the following conditions exist:

- The job step is currently executing.
- The job step is currently being placed on the reader queue or is being copied by the spool file copy program in another partition.
- The QCOPY-NO parameter was specified on the JOB OCL statement for that job.

When a job is being copied from the reader queue, that job appears on a display of the queue with an indication that it is being copied by $QCOPY. However, system commands that normally pertain to that job do not affect it.

If the output file is tape, a default block length of 960 (ten records per block) is assumed unless a block length for the file is specified on the OCL FILE statement. The minimum partition size when using tape is 10K bytes plus twice the tape block length rounded upward to the nearest 2K.

When REMOVE-YES is specified, jobs are not removed from the queue until all jobs have been successfully copied. However, jobs are removed from the queue before the next control statement is read. Therefore, jobs being copied are not lost if the copy function does not terminate normally. But jobs will be lost if $QCOPY is canceled after the jobs have been removed.

When REMOVE-ONLY is specified, jobs are removed from the queue without being copied. The READERQ file is not used for this function; however, the OUTPUT parameter must be specified.

Comments are not allowed on the COPYRDRQ control statement unless at least one parameter is included.

Note: When input to the reader queue is from the system input device and a cancel option is taken to an error message, the JOB statements and the /. records used to terminate the copy reader queue function also terminate the flush of the system input device that occurs after the cancel option. The LOAD, CALL, and /& statements can also terminate the flush if the spool file copy program is executed when the system is not in job mode.

Read Control Statements from a File (COPYCTRL)

The COPYSP, COPYCTRL, AUTHORIZE, and CLASSIFY control statements cannot be specified in the CONTROL file.

Comments are not allowed on the COPYCTRL control statement if there are no parameters included.

No more than 254 control statement records are allowed in the CONTROL file. The CONTROL file can be any record length from 1 through 128.

The COPYCTRL control statement is valid only when the spool file copy program is executed under the communications control program.

Copy a Display of the Status of the Spool Queues (DISPLAY)

Comments are not allowed on the DISPLAY control statement if there are no parameters included.

All queues (read, punch, and print) are copied if OUTPUT-FILE is specified and the Q-parameter is not included. The status of the reader queue is copied first, followed by the print queue, then the punch queue. Each queue is displayed in its entirety. The first record is the queue heading record, which is followed by the queue status records and an END OF QUEUE record.

When the priority limit is less than 5 for jobs placed on the active spool file reader queue, a display of the active spool file reader queue contains a line at the top of the screen that indicates the priority limit.

The print queue is displayed when OUTPUT-TERMINAL is specified and the Q-parameter is not included on the control statement.

When OUTPUT-FILE is specified, the record length is 40 bytes. If the output file is tape, a default block length of 400 (ten records per block is assumed unless a block length for the file is specified on the OCL FILE statement. The minimum partition size when using tape is 10K bytes plus twice the tape block length rounded upward to the nearest 2K.

If the output file is disk, the file can be shared only when it is closed.

When the spool file copy program is executed under SCP, OUTPUT-TERMINAL is invalid and the default is OUTPUT-FILE. When the program is executed under CCP, OUTPUT-TERMINAL is the default.

For more information about queue display, refer to the *IBM System/3 Model 15 User's Guide to Spooling,* GC21-7632.

This page intentionally left blank.

Restore Print or Punch Queue Records From a File (RESTORE)

The restore function requires a minimum partition size of 12K bytes.

The RESTORE file must be a disk or tape file that was created by either the COPYPRTQ or the COPYPCHQ function of $QCOPY with OUTPUT-FILE or OUTPUT-ADD specified.

The RESTORE file record length is used to determine which device the records in the file are for. The following chart shows the valid input record lengths and the associated devices:

Input Record Length	Output Device
82	1442
83	2560
98	5424
134	1403

If the RESTORE file is a tape file and the tape does not contain a standard label or the REEL-BLP parameter is specified, then both the block length and the record length must be specified on the OCL FILE statement for the file. The minimum partition size when using tape is 12K bytes plus twice the tape block length rounded upward to the nearest 2K.

The RESTORE file can contain job steps that have header records, job steps that do not have any header records, or both job steps that have header records and job steps that do not have header records. Also, print job steps can be VLR, FLR, or a mix of VLR and FLR steps.

Job steps that have header records can be selectively restored by jobname, stepname, and forms type (print) or card type (punch). Additionally, a message is issued if STOP is requested and the records are being restored to a spooled device in a nonspooled partition (OUTPUT-PRINT or OUTPUT-PUNCH).

Job steps without header records cannot be restored by jobname, stepname, and forms type (print) or card type (punch) even though these parameters are specified on the RESTORE statement. Also, no message can be issued between steps, when a forms type changes, or when a card type changes.

If the RESTORE function with OUTPUT-PRINT or OUTPUT-PUNCH is being executed in a partition that is being spooled, and the print or punch output is for the same print or punch device being spooled, then the records being printed or punched are intercepted by spool and placed on the queue as one job step, under the jobname and stepname used when the spool file copy was loaded.

When a job step is restored from a file to a spool queue (OUTPUT-SPOOL), that job step must have a header record. When placed on the queue, the job step will have the same jobname, stepname, forms type (print) or card type (punch), priority, hold status, and keep status as it had originally. A restored job step is considered to be part of the same job as the preceding restored job step only if both job steps have the same jobname and the same priority. Otherwise, the job step is considered as a different job.

Comments are not allowed on the RESTORE statement unless at least one parameter is included.

Note: If punch records are being restored from a file to the spool punch queue (OUTPUT-SPOOL), no check is made to verify that the punch records are for the same device as the spool punch device; unpredictable results can occur if the devices are not the same.

Copy Selected Job Steps From One Spool File to Another (COPYQ)

A job step on a queue cannot be copied if any of the following conditions exist:

- The job step is currently executing.
- The job step is being used by spool or copied by the spool file copy program in another partition.
- A REUSE command was specified for the job step.
- A QCOPY-NO parameter is specified on the OCL statement pertaining to the desired queue.

All available job steps on the queue (including those that are currently held) are copied when COPYQ is specified without any of the following parameters: JOBN, STEPN, FORMSNO. CARDNO, PRIORITY, PARITION. When a job step is being copied, that job step appears on a display of the queue with an indication that it is being copied by $QCOPY. However, system commands that normally pertain to that job step do not affect it.

When the COPYQ function is executed under control of the SCP (system control program), the spool file copy program can create a new spool file on the main data area designated on the TO parameter. However, the new spool file is created only if one does not exist prior to executing the COPYQ function under control of the SCP. If the spool file is created by $QCOPY, it will be the same size and track group size as the file specified on the FROM parameter.

A new spool file can be created without any jobs on the queues by specifying a JOBN parameter that does not exist on the spool file specified by the FROM parameter.

If the COPYQ function is executed under control of the communications control program (CCP), there must be a spool file on the main data area specified by the TO parameter.

When REMOVE-YES is specified, jobs/job steps are not removed from the queue until all jobs/job steps have been successfully copied. However, jobs/job steps are removed from the queue before the next control statement is read. Therefore, jobs/job steps are not lost if the copy function does not terminate normally. But jobs/job steps will be lost if $QCOPY is canceled after the jobs/job steps have been removed.

Note: When punch queue job steps are copied from one spool file to another, $QCOPY does not verify that the punch records on each spool file are for the same punch device; unpredictable results can occur if the devices are not the same. Similarly, when reader queue jobs are copied from one spool file to another, unpredictable results can occur if the spool reader record length is not the same for each spool file.

MAINTAIN THE AUTHORIZ FILE (AUTHORIZE)

The AUTHORIZE control statement is valid only when the spool file copy program is executed under the system control program; however, it is not valid in a procedure.

Authorization information is maintained in a main data area disk file. The record length of the file is 256 bytes. Each logical record in the file contains 16 authorization records, so that a 1-track file will hold 768 passwords and their associated authorization information. The file must originally be created as an empty direct file. This can be done with $COPY with the following OCL and control statements:

```
// LOAD $COPY,F1
// FILE NAME-COPYIN,UNIT-READER
// FILE NAME-COPYO,UNIT-D1,PACK-D1D1D1,
//       TRACKS-1,LABEL-AUTHORIZ,RETAIN-P
// RUN
// COPYFILE OUTPUT-FILE,LENGTH-256
// OUTDM DATAMGMT-DIRECT
// END
/*
```

Before $CNFIG is used to require $QCOPY authorization, a record should be placed in the AUTHORIZ file that contains a password allowing authorization changes. For information concerning the creation of an authorization record, refer to *Create Authorization Record.*

Authorization records in the file are created, updated, or deleted based on the authorization change records that are read from the system input device. These records must follow the AUTHORIZE control statement. (If $QCOPY user authorization is required in the system when the AUTHORIZ file is being changed, a record with a password in columns 1 through 4 must precede the authorization change records in the system input device; the password must authorize changes to the AUTHORIZ file.)

Authorization change records have the following format:

Columns 1-4	Old password
Columns 5-8	New password
Columns 9-12	Old authroizations
Columns 13-16	New authorizations
Columns 17-24	Old user ID
Columns 25-32	New user ID

The old password is the password as it currently exists; the new password is what the password is to be changed to. Passwords must consist of four letters of the alphabet and/or numeric digits in any combination (but not $, #, @, or any special characters).

When a listing is requested (LIST-YES), the user ID is used to associate a user (for example, by last name) with a password. This association is used for informational purposes only and is not used by $QCOPY in determining whether a user is authorized for specific functions. If this association is not desired, the old and new user ID fields must be blank.

Authorization Fields

The authorizations field indicates which functions of $QCOPY are authorized by the password. Each position of the authorizations field has a specific meaning. Within each position the values 0 through 7 correspond with authorization for specific functions of $QCOPY.

Position 1

Position 1 controls the authorization for changing the AUTHORIZ file, the authorization for assigning class numbers to programs, and the authorization for the various classes of programs that can be executed in a batch partition. The following chart relates these authorizations with the values (0-7) that can be specified in position 1.

	Values for Position 1							
Functions	0	1	2	3	4	5	6	7
Change the AUTHORIZ file	N	N	N	N	Y	Y	Y	Y
Assign class numbers to programs	N	N	N	N	Y	Y	Y	Y
Execute batch programs of class 0	Y	Y	Y	Y	Y	Y	Y	Y
Execute batch programs of classes 0 and 1	N	Y	Y	Y	N	Y	Y	Y
Execute batch programs of classes 0, 1, and 2	N	N	Y	Y	N	N	Y	Y
Execute batch programs of classes 0, 1, 2, and 3	N	N	N	Y	N	N	N	Y

For example, a value of 2 will allow the execution of batch programs in classes 0, 1, and 2 (not 3) and will not allow changes to the AUTHORIZ file, nor will it allow the assigning of class numbers to programs. A value of 5 will allow the execution of batch programs in classes 0 and 1 (not 2 or 3), and will allow changes to the AUTHORIZ file, and will also allow the assigning of class numbers to programs. Executing a batch program assumes the authorization to place jobs on the reader queue (see position 2, values 4, 5, 6, and 7).

Position 2

Position 2 controls the authorization for access to the reader queue. Authorization may be granted for copying jobs from the queue, removing jobs from the queue, and/or copying jobs to the queue. The following chart relates these authorizations with the values (0-7) that can be specified in position 2.

	Values for Positions 2, 3, and 4							
Functions	0	1	2	3	4	5	6	7
Copy jobs from the queue	N	Y	N	Y	N	Y	N	Y
Remove jobs from the queue	N	N	Y	Y	N	N	Y	Y
Copy jobs to the queue	N	N	N	N	Y	Y	Y	Y

For example, a value of 0 prevents any access to the reader queue. A value of 1 allows copying from the queue, but does not allow removing from the queue or copying to it. A value of 7 allows copying from, removing from, and copying to the queue. (A value of 4 or greater must be specified in position 2 when any value (other than 0 or 4) is specified in position 1.)

$QCOPY Control Statements		Program Class	Value in Authorization Field: Position 1 (0 1 2 3 4 5 6 7)	Position 2 (0 1 2 3 4 5 6 7)	Position 3 (0 1 2 3 4 5 6 7)	Position 4 (0 1 2 3 4 5 6 7)
// COPYSP				5 7	5 7	5 7
// COPYPRTQ	(REMOVE-NO)				1 3 5 7	
// COPYPRTQ	REMOVE-ONLY				2 3 6 7	
// COPYPRTQ	REMOVE-YES				3 7	
// COPYPCHQ	(REMOVE-NO)					1 3 5 7
// COPYPCHQ	REMOVE-ONLY					2 3 6 7
// COPYPCHQ	REMOVE-YES					3 7
// COPYRDRQ	OUTPUT-FILE/ADD,(REMOVE-NO)			1 3 5 7		
// COPYRDRQ	OUTPUT-FILE/ADD,REMOVE-ONLY			2 3 6 7		
// COPYRDRQ	OUTPUT-FILE/ADD,REMOVE-YES			3 7		
// COPYRDRQ	(INPUT-FILE/READER/TERMINAL)	0	0 1 2 3 4 5 6 7	4 5 6 7		
// COPYRDRQ	(INPUT-FILE/READER/TERMINAL)	1	1 2 3 5 6 7	4 5 6 7		
// COPYRDRQ	(INPUT-FILE/READER/TERMINAL)	2	2 3 6 7	4 5 6 7		
// COPYRDRQ	(INPUT-FILE/READER/TERMINAL)	3	3 7	4 5 6 7		
// RESTORE	(to print queue)				4 5 6 7	
// RESTORE	(to punch queue)					4 5 6 7
// COPYQ	Q-WQ,(REMOVE-NO)				5 7	
// COPYQ	Q-WQ,REMOVE-YES				7	
// COPYQ	Q-PQ,(REMOVE-NO)					5 7
// COPYQ	Q-PQ,REMOVE-YES					7
// COPYQ	Q-RQ,(REMOVE-NO)	0	0 1 2 3 4 5 6 7	5 7		
// COPYQ	Q-RQ,(REMOVE-NO)	1	1 2 3 5 6 7	5 7		
// COPYQ	Q-RQ,(REMOVE-NO)	2	2 3 6 7	5 7		
// COPYQ	Q-RQ,(REMOVE-NO)	3	3 7	5 7		
// COPYQ	Q-RQ,REMOVE-YES	0	0 1 2 3 4 5 6 7	7		
// COPYQ	Q-RQ,REMOVE-YES	1	1 2 3 5 6 7	7		
// COPYQ	Q-RQ,REMOVE-YES	2	2 3 6 7	7		
// COPYQ	Q-RQ,REMOVE-YES	3	3 7	7		
// AUTHORIZE			4 5 6 7			
// CLASSIFY			4 5 6 7			

Figure 4-94.1. Authorization Function Values for QCOPY Functions

Positions 3 and 4

Position 3 controls the authorization for access to the print queue, and position 4 controls the authorization for access to the punch queue. The authorizations granted by the values 0 through 7 for the print and punch queues are the same as for the reader queue.

A value must be specified in each position of the authorization field. Figure 4-94.1 can be used to assist in selecting the values for each position based on the functions you want to allow. Within each position, any of the values specified on the chart authorizes the corresponding control statement. (Positions that do not have a value specified on the chart are of no consequence to the specific control statement and may contain any value from 0 through 7.)

For example, a value of 1, 3, 5, or 7 is required in position 3 to authorize // COPYPRTQ REMOVE-NO; the authorizations field may be coded xx1x, xx3x, xx5x, or xx7x. The 1, 3, 5, and 7 values in position 3 each authorize other print queue functions; so the right value must be chosen, based on what functions are to be allowed and what other functions are not to be allowed.

The // COPYSP statement requires values of 5 or 7 in positions 2, 3, and 4. Thus, authorization fields (for example, x555, x575, x777) all allow // COPYSP. (Note that these values also allow other functions that involve copying to or copying from any spool queue.)

Create Authorization Record

To create a new authorization record in the file, an authorization change record that has blanks in the old password field must be read in the system input device. If the new password is unique, an authorization record with the new password, user ID, and authorizations is placed in the file.

Change Authorization Record

To change an authorization record in the file, an authorization change record must be read in the system input device with old password, old authorizations, and old user ID fields that match the record being updated in the file. The record is updated with the new password, new authorizations, and new user ID. Passwords, authorizations, and/or user IDs in any combination can be updated in this manner. If a particular field is not to be changed, the old and new fields must match.

Delete Authorization Record

To delete an authorization record in the file, an authorization change record must be read in the system input device with old password, old authorizations, and old user ID fields that match the record being deleted in the file; the new password field must be blank. The old password becomes invalid when the authorization record is deleted.

The end of the authorization change records is indicated by the next $QCOPY control statement read in the system input device. Before this statement is processed, a listing of the information in the AUTHORIZ file is printed on the system printer if LIST-YES was specified on the AUTHORIZE control statement.

When an error is detected in an authorization change record, a message with cancel and continue options is displayed on the system console.

An OCL FILE statement describing the authorization file (NAME-AUTHORIZ) must be included in the OCL when the AUTHORIZE control statement is to be used. The file cannot be shared when referenced by this control statement.

Comments are not allowed on the AUTHORIZE control statement if there are no parameters included.

If $QCOPY (with user authorization required) is executed in a batch partition as a result of being put on the reader queue from CCP, functions that were not authorized under CCP are not allowed in the batch partition.

Figure 4-94.2 shows a sample listing of the contents of the AUTHORIZ file.

ASSIGN A CLASS NUMBER TO A PROGRAM (CLASSIFY)

The CLASSIFY statement is valid only when the spool file copy program is executed in a batch partition.

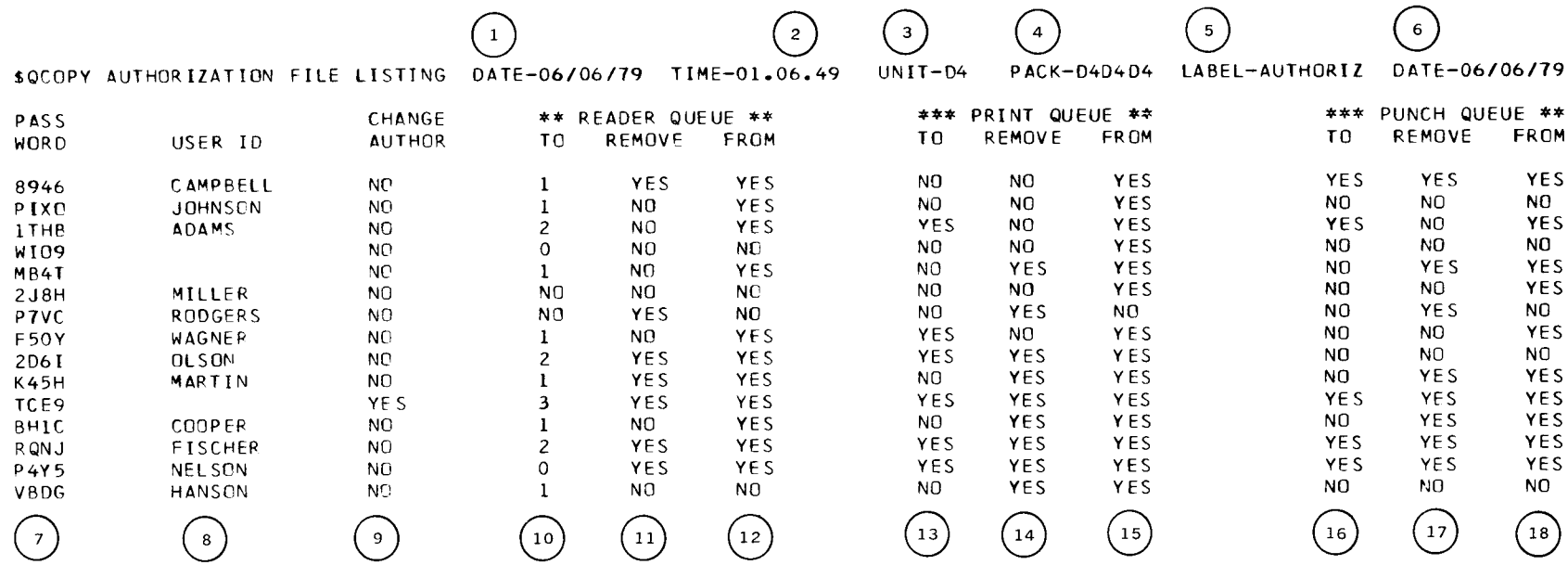
It may not be desirable for every user (with authorization to place jobs on the reader queue) to be able to execute every program in the system. Program classes can be used to restrict individual users of $QCOPY under CCP to only a subset of the programs because users can execute only those programs in the class(es) for which they are authorized. Four program classes (0 through 3) are available. The higher the program class number, the higher the authorization required to execute the program in the batch partition. Program class authorization levels are also numbered 0 through 3. A user can execute a program if the class number of the program is less than or equal to the authorization level of the user.

Program classes are defined in whatever way best suits the needs of the user. Programs should be classified accordingly, and individual users should be assigned authorization levels that allow them to execute only the appropriate programs. The following considerations should be observed:

- A user with program class authorization level 3 may execute any program in the system.
- A program of class 0 may be executed by any user who is authorized to place jobs on the reader queue.
- All programs supplied by IBM (each release) and all previously unclassified programs default to class 0.

A class number assigned to an R-module has no meaning with respect to that R-module except that the class number is transferred to every program (O-module) compiled using that R-module. When a program is compiled, it is assigned a class number equal to the highest class number of all the R-modules included during link-editing. Thus, a program can be assigned a class number that will be preserved through recompilation if that class number is assigned to one of the R-modules to be included.

Program class numbers are kept in the module attributes in the library directory and are shown in a $MAINT listing of the O-library directory. (See *$MAINT* in Part 4 of this manual.)



$QCOPY AUTHORIZATION FILE LISTING (1) DATE-06/06/79 (2) TIME-01.06.49 (3) UNIT-D4 (4) PACK-D4D4D4 (5) LABEL-AUTHORIZ (6) DATE-06/06/79

(7) PASS WORD	(8) USER ID	(9) CHANGE AUTHOR	** READER QUEUE ** (10) TO	(11) REMOVE	(12) FROM	*** PRINT QUEUE ** (13) TO	(14) REMOVE	(15) FROM	*** PUNCH QUEUE ** (16) TO	(17) REMOVE	(18) FROM
8946	CAMPBELL	NO	1	YES	YES	NO	NO	YES	YES	YES	YES
P1X0	JOHNSON	NO	1	NO	YES	NO	NO	YES	NO	NO	NO
1THB	ADAMS	NO	2	NO	YES	YES	NO	YES	YES	NO	YES
WI09		NO	0	NO	NO	NO	NO	YES	NO	NO	NO
MB4T		NO	1	NO	YES	NO	YES	YES	NO	YES	YES
2J8H	MILLER	NO	NO	NO	NO	NO	NO	YES	NO	NO	YES
P7VC	RODGERS	NO	NO	YES	NO	NO	YES	NO	NO	YES	NO
F50Y	WAGNER	NO	1	NO	YES	YES	NO	YES	NO	NO	YES
2D6I	OLSON	NO	2	YES	YES	YES	YES	YES	NO	NO	NO
K45H	MARTIN	NO	1	YES	YES	NO	YES	YES	NO	YES	YES
TCE9		YES	3	YES	YES	YES	YES	YES	YES	YES	YES
BH1C	COOPER	NO	1	NO	YES	NO	YES	YES	NO	YES	YES
RQNJ	FISCHER	NO	2	YES	YES	YES	YES	YES	YES	YES	YES
P4Y5	NELSON	NO	0	YES	YES	YES	YES	YES	YES	YES	YES
VBDG	HANSON	NO	1	NO	NO	NO	YES	YES	NO	NO	NO

Figure 4-94.2. Sample Listing of the AUTHORIZ File

(1) Date of the authorization file listing

(2) Time of the authorization file listing (if time-of-day was selected during Sysgen)

(3) Disk unit of the authorization file

(4) Disk pack containing the authorization file

(5) Label of the authorization file

(6) Date of the authorization file

(7) Passwords in the authorization file

(8) User IDs associated with passwords (blanks indicate no user ID specified)

(9) Authorization to change the authorization file

(10) Program class authorization level for placing jobs on the reader queue (NO indicates no authorization to place jobs on the reader queue)

(11) Authorization to remove jobs from the reader queue

(12) Authorization to copy jobs from the reader queue

(13) Authorization to copy job steps to the print queue

(14) Authorization to remove job steps from the print queue

(15) Authorization to copy job steps from the print queue

(16) Authorization to copy job steps to the punch queue

(17) Authorization to remove job steps from the punch queue

(18) Authorization to copy job steps from the punch queue

Page of GC21-5162-1
Issued 28 September 1979
By TNL: GN21-5674

OCL CONSIDERATIONS

The following OCL statements are required to load the $QCOPY system service program:

```
// LOAD $QCOPY,code
// RUN
```

The code you supply depends on the location of the simulation area containing the spool file copy program. Possible codes are F1, R1, F2, R2.

EXAMPLES

Figure 4-95 through 102.2 are examples of the OCL statements and control statements needed to:

- Copy an entire spool file from disk to magnetic tape
- Print selected job steps from the print queue
- Punch selected job steps from the punch queue
- Copy a file to the reader queue
- Copy selected jobs from the reader queue
- Copy a display of the reader queue to a file
- Restore a print or punch queue file to the spool print or punch queue
- Copy a job step from one spool file to another spool file
- Change the AUTHORIZ file
- Assign a class number to a program

```
1   4   8   12  16  20  24  28  32  36  40  44  48
// LOAD $QCOPY,F1
// FILE NAME-$SPOOL,UNIT-T1,REEL-NL,END-REWIND
// RUN
// COPYSP FROM-D4,TO-TAPE
// END
```

Explanation:

- The spool file copy program is loaded from F1.

- Output file (OCL sequence):
  - The name of the output file is $SPOOL (NAME-$SPOOL).
  - The output file is copied to tape drive 1 (UNIT-T1).
  - The output file is copied to an unlabeled tape (REEL-NL).
  - The tape is rewound at EOJ (END-REWIND).

- Control statement:
  - The entire spool file is copied (COPYSP).
  - The area containing the spool file is D4 (FROM-D4).
  - The spool file is copied to magnetic tape (TO-TAPE).

Figure 4-95. OCL and Control Statements to Copy an Entire Spool File from Disk to Magnetic Tape

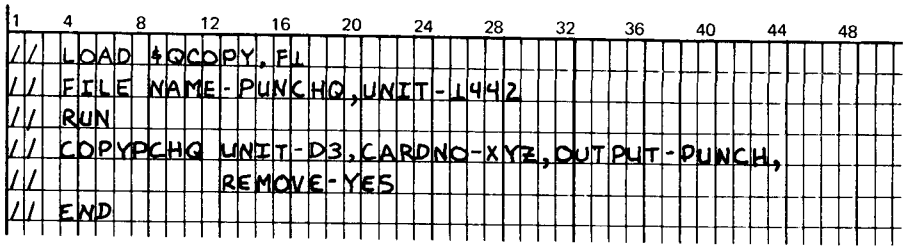
```
1   4   8   12  16  20  24  28  32  36  40  44  48
// LOAD $QCOPY,F1
// RUN
// COPYPRTQ UNIT-D3,FORMSNO-ABC,JOBN-BILLINGS,
//          STEPN-LISTING,OUTPUT-PRINT
// END
```

Explanation:

- The spool file copy program is loaded from F1.

- Control statement:
  - Job steps are copied from the print queue of the spool file (COPYPRTQ).
  - The area containing the spool file is D3 (UNIT-D3).
  - Only those job steps on the print queue with the jobname BILLINGS (JOBN-BILLINGS), stepname LISTING (STEPN-LISTING), and forms type ABC (FORMSNO-ABC) are copied.
  - The output is printed (OUTPUT-PRINT).

Figure 4-96. OCL and Control Statements to Print Selected Job Steps from the Print Queue



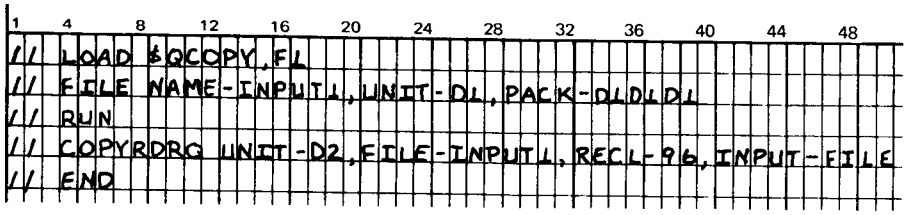
Explanation:

- The spool file copy program is loaded from F1.

- Output file (OCL sequence):
  - The name that identifies the file is PUNCHQ (NAME-PUNCHQ).
  - The output file is contained in the 1442 (UNIT-1442).

- Control statement:
  - Job steps are copied from the punch queue of the spool file (COPYPCHQ).
  - The area containing the spool file is D3 (UNIT-D3).
  - Only those job steps with card type XYZ are punched (CARDNO-XYZ).
  - The output is punched in the same format as that punched by the spool punch writer (OUTPUT-PUNCH).
  - If multiple copies are specified, the number is reduced by 1. Otherwise, the job step is removed from the punch queue after it is copied (REMOVE-YES).

Figure 4-97. OCL and Control Statements to Punch Selected Job Steps from the Punch Queue



Explanation:

- The spool file copy program is loaded from F1.

- Input file (OCL sequence):
  - The name that identifies the input file is INPUT1 (NAME-INPUT1).
  - The area that contains the file is D1 (UNIT-D1). Its name is D1D1D1 (PACK-D1D1D1).

- Control statement:
  - Jobs are placed on the spool file reader queue (COPYRDRQ).
  - The area that contains the spool file is D2 (UNIT-D2).
  - The file named INPUT1 (FILE-INPUT1) contains the jobs that are placed on the reader queue.
  - The length of the records on the spool reader queue is 96 bytes (RECL-96).
  - The source of the records is identified as a file (INPUT-FILE).

Figure 4-98. OCL and Control Statements to Copy a File to the Reader Queue

```
1   4   8   12  16  20  24  28  32  36  40  44  48  52
//  LOAD $QCOPY,F1
//  FILE NAME-READERQ,UNIT-D1,PACK-D1D1D1,TRACKS-10
//  RUN
//  COPYRDRQ UNIT-D2,OUTPUT-FILE
//  END
```

Explanation:

- The spool file copy program is loaded from F1.
- Output file (OCL sequence):
  - The name that identifies the file is READERQ (NAME-READERQ).
  - The area that contains the file is D1 (UNIT-D1). The name of the area is D1D1D1 (PACK-D1D1D1).
- Control statement:
  - Jobs are copied from the spool file reader queue (OUTPUT-FILE).
  - The area that contains the spool file is D2 (UNIT-D2).

Figure 4-99. OCL and Control Statements to Copy a File from the Reader Queue

```
1   4   8   12  16  20  24  28  32  36  40  44  48
//  LOAD $QCOPY,F1
//  FILE NAME-DISPLAYQ,UNIT-D2,TRACKS-1,PACK-D2D2D2
//  RUN
//  DISPLAY Q-RQ
//  END
```

Explanation:

- The spool file copy program is loaded from F1.
- Output file (OCL sequence):
  - The name that identifies the file is DISPLAYQ (NAME-DISPLAYQ).
  - The area that contains the file is D2 (UNIT-D2). Its name is D2D2D2 (PACK-D2D2D2).
  - The file occupies 1 track (TRACKS-1).
- Control statement:
  - A display of the reader queue is copied to the output file (Q-RQ).

Figure 4-100. OCL and Control Statements to Copy a Display of the Reader Queue to a File

```
1   4   8   12  16  20  24  28  32  36  40  44  48  52
// LOAD $QCOPY,F1
// FILE NAME-RESTORE,UNIT-D1,PACK-D1D1D1
// RUN
// RESTORE UNIT-D2
// END
```

Explanation:

- The spool file copy program is loaded from F1.
- Input file (OCL sequence):
  - The name that identifies the file is RESTORE (NAME-RESTORE).
  - The area that contains the file is D1 (UNIT-D1). The name of the area is D1D1D1 (PACK-D1D1D1).
- Control statement:
  - The print or punch queue records contained in the file are restored to the respective spool print or punch queue (RESTORE).
  - The spool file that contains the queue to which the records are to be restored is located on main data area D2 (UNIT-D2).

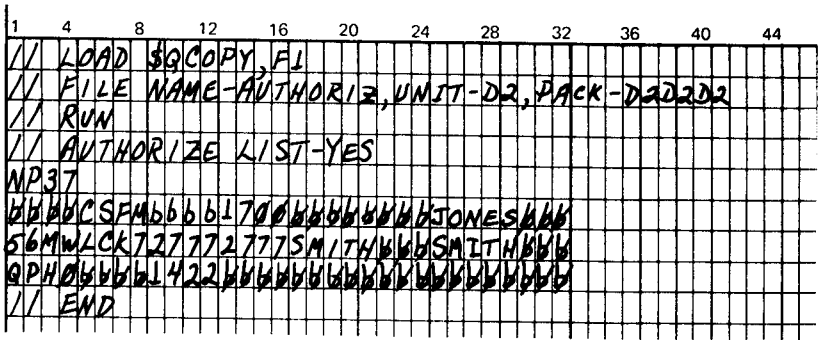
Figure 4-101. Restore Print or Punch Queue Records

```
1   4   8   12  16  20  24  28  32  36
// LOAD $QCOPY,F1
// RUN
// COPYQ FROM-D1,TO-D2,Q-WQ
// END
```

Explanation:

- The spool file copy program is loaded from F1.
- Control statement:
  - The job steps on the print queue (Q-WQ) are copied from the spool file on D1 (FROM-D1) to the print queue of the spool file on D2 (TO-D2).

Figure 4-102. Copy the Print Queue from One Spool File to Another



Explanation:

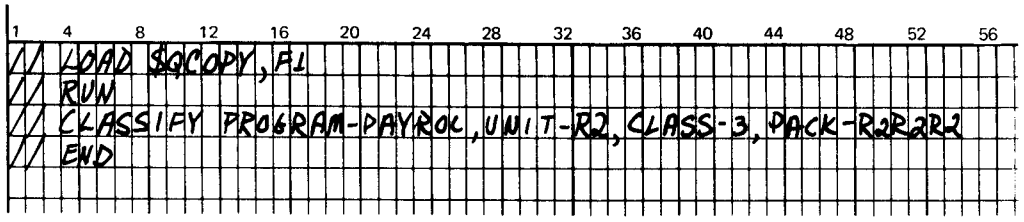
- The spool file copy program is loaded from F1.

- Input and update file (OCL sequence):
  - The name that identifies the file is AUTHORIZ (NAME-AUTHORIZ).
  - The area that contains the file is D2 (UNIT-D2).
  - The name of the area is D2D2D2 (PACK-D2D2D2).

- Control statement:
  - The authorization change records follow the control statement (AUTHORIZE) in the system input device and are used to change the AUTHORIZ file.
  - A listing of the contents of the AUTHORIZ file is printed on the system print device when the changes to the file are complete (LIST-YES).

- Authorization change records:
  - The first record (NP37) is a password that allows changes to be made to the AUTHORIZ file.
  - The second record contains a blank in each of the four leftmost columns to indicate that CSFM is to be a new password in the file. The authorization code associated with the password is 1700 and the user ID is JONES.
  - The third record indicates that the password (56MW) in the file is to be changed to LCK7. The authorization code (2777) and the user ID (SMITH) stay the same.
  - The fourth record contains a blank in each of the 5, 6, 7, and 8 positions to indicate that the password (QPH0) is to be deleted from the file.

Figure 4-102.1. Change the AUTHORIZ File



Explanation:

- The spool file copy program is loaded from F1.
- Control statement:
  - The program PAYROL (PROGRAM-PAYROL) is to be assigned a class number.
  - The PAYROL program is in the O library ($QCOPY assumes the LIBRARY-O default). The O library is in simulation area R2 (UNIT-R2).
  - The PAYROL program is to be assigned class number 3 (CLASS-3).
  - The name of the simulation area on R2 is R2R2R2 (PACK-R2R2R2).

Figure 4-102.2. Assign a Class Number to a Program

USING THE SPOOL FILE COPY PROGRAM UNDER CCP

The system cannot execute the spool file copy program under CCP with the name $QCOPY. Use the library maintenance program ($MAINT) or the rename $QCOPY prompt during system generation to change the program name so that the first character is not a dollar sign ($).

If $QCOPY is to be executed under CCP, the renamed modules must reside on the CCP program pack.

When the spool file copy program is executed as a CCP task, the files accessed for the various functions (for example, the control file, display queue file, print queue file, reader queue file, restore file, and the punch queue file) must be disk files

Output files (DISPLAYQ, PRINTQ, PUNCHQ, and READERQ) can be accessed either as consecutive output or as consecutive add. If accessed as consecutive add, the files can be shared between batch and CCP partitions when the files are closed instead of having to wait until CCP is shut down. The consecutive add access appears to be consecutive output because the records are added from the beginning of the file. If it is desirable to add records from the logical end of the file, rather than from the beginning of the file, OUTPUT-ADD can be specified on the appropriate control statement. The add access from the beginning of the file will not cause any messages to be issued that warn of overlaying existing records in the file during CCP start-up.

Certain functions of the spool file copy program require more than 10K bytes of main storage. (See Figure 4-94 for the storage requirements of the various functions.) The TASKSIZE parameter of the PROGRAM statement in the CCP assignment set must be used to specify the required storage size when it exceeds 10K. For a description of the TASKSIZE parameter, refer to the *IBM System/3 Model 15 Communications Control Program System Reference Manual,* GC21-7620.

Program Request

When the spool file copy program is requested under CCP, data may be supplied with the program request. A maximum of two fields, separated by a comma, can be entered. The first field of data is the name of a file from which the control statements for the spool file copy program will be read. If this field is entered, the length of the file name can be from 1 through 8 characters. A FILE statement with that name must be included in the CCP startup OCL.

The second field of data is a 4-character password that grants the user authorization for various functions of the spool file copy program. The password is required if the configuration record program option is in effect requiring $QCOPY user authorization. (See $CNFIG in Part 4 of this manual.) The following examples of program request show how the fields of program data must be entered:

QCOPY	No password and no CONTROL file name given.
QCOPY filename	CONTROL file name given with no password.
QCOPY ,pswd	Password given with no CONTROL file name.
QCOPY filename,pswd	CONTROL file name given with password.

When requested under CCP from the system console or from a terminal that is neither a 3275 Model 1 or 2, nor a 3277 Model 1 or 2, the spool copy program reads control statements only from a file. The name of that file may be given as the first field of data with the program request. If a file name is not supplied, a default name of CONTROL is assumed.

The following functions of the spool file copy program are not supported when requested under CCP from the system console or from a terminal that is not a 3275 or a 3277:

- Displaying the spool queues to the requesting terminal.
- Entering jobs onto the reader queue from the requesting terminal.
- Issuing messages to the requesting terminal. These messages will go instead to the system console.

For a complete list of terminals supported by CCP, refer to the *IBM System/3 Model 15 Communications Control Program System Reference Manual,* GC21-7620.

User Authorization

If the $QCOPY user authorization option of $CNFIG is in effect, authorization is required for the use of the spool file copy program under CCP. The functions that $QCOPY can perform are associated with the password entered with the program request. Authorization can be granted for the following functions:

- Request the spool file copy program under CCP.
- Access the reader queue under CCP.
- Access the print queue under CCP.
- Access the punch queue under CCP.
- Change the $QCOPY authorizations and assign class numbers to programs.

Authorization for access to each queue is further granted for the following functions:

- Copy jobs and/or job steps to the queue.
- Copy jobs and/or job steps from the queue.
- Remove jobs and/or job steps from the queue.

Authorization for copying jobs onto the reader queue is further granted for the purpose of executing different classes of programs in the batch partition. A program cannot be executed if it is classified higher than the program class authorization level associated with the password that was given with the program request.

Passwords are maintained in the AUTHORIZ file. When the spool file copy program is first loaded under CCP, this file is opened with a direct get (DG) access and is searched for the password given in the program request data. If the password is not found in the file, the user is not authorized to request the spool file copy program. If the password is found, the authorization information associated with it is used to verify that the user is authorized for the requested functions. Any time the user attempts to execute an unauthorized function, the spool file copy program terminates with a code 41. An OCL FILE statement with NAME-AUTHORIZ must be specified in the startup OCL for CCP if $QCOPY user authorization is required.

Using the Spool File Copy Program from a Terminal

When requested from a 3275 or 3277 terminal, the spool file copy program erases the screen for each message. An option of the configuration record program (see $CNFIG in Part 4 of this manual) causes the screen to not be erased for the following functions:

- Prompting for control statements
- Diagnosing errors
- Prompting for data to be entered on the reader queue

The screen is always erased for the following conditions:

- The first write to the screen after the spool file copy program is loaded
- Going to and from ENTER RDRQ DATA mode (// COPYRDRQ INPUT-TERMINAL)
- During DISPLAY mode (// DISPLAY)

When the spool file copy program is requested under CCP from a 3275 or a 3277 terminal and a file name is supplied in the first field of program data, control statements are read from that file. The requesting terminal can be used for displaying the queues, entering jobs on the reader queue, or issuing messages. Control statements will not be read from the terminal unless the reading of the CONTROL file is terminated by a control statement error. Otherwise, when an END statement is read from the file or when end of file is detected, the spool file copy program goes to end of job.

When the spool file copy program is requested from a 3275 or a 3277 terminal and no control file name is specified in the first field of program data, the UQ SP CS ENTER CONTROL STATEMENT message is sent to the terminal and the cursor is set to position 1 of line 1. The terminal operator then keys the program control statement into the first 80 positions. The format of the control statements and the rules of continuation are the same as previously described in this manual. The ENTER key on the terminal is pressed to complete the entry of the program control statement. The control statement is then logged in the system history area.

This page intentionally left blank.

Responding to Error Messages

The spool file copy program diagnoses errors in control statements as well as error conditions which may occur during the execution of the program.

If an error is found in a control statement, the UQ SP XX CONTROL STATEMENT ERROR message is issued to the terminal. The response to this message is to correct and reenter the control statement. No option is required.

If an error occurs during execution of the program, the UQ SP XX ENTER RECOVERY OPTION YYYY message is issued to the terminal for which the spool file copy program is executing. The XX indicates the kind of error. The YYYY indicates the allowable options to the message. The following are the allowable options to messages and their meaning:

Option	Meaning
0	Close the function (and the associated files) and process the next control statement
1	Retry the function
2	Close the function (and the associated files) and exit to end-of-job
3	Immediate exit to end-of-job (close the associated files)

The error messages are displayed on the screen. When a message requires a decision or action, you respond via the keyboard by entering the option desired and then pressing the ENTER key.

The message ID (UQ SP XX) which identifies the error that occurred, is described in the *IBM System/3 Model 15 System Messages,* GC21-5076 and the *IBM System/3 Communications Control Program Messages Manual,* GC21-5170.

Placing Jobs on the Reader Queue from a Terminal

The spool file copy program allows the terminal operator to key OCL or data onto the spool reader queue. When INPUT-TERMINAL is specified on the COPYRDRQ control statement, the program sends the UQ SP RQ ENTER RDRQ DATA message to the terminal and positions the cursor to line 1. The terminal operator can then key an 80-byte OCL or data statement. The terminal operator then presses the ENTER key to enter the statement into the system. Jobs are placed on the reader queue in a manner similar to that of the spool reader.

To terminate the copy-to-reader-queue function, two consecutive /. statements are required.

Displaying the Spool Queues

When the spool file copy program is performing the display function, the display can be directed to the terminal (OUTPUT-TERMINAL) that made the program request. The display begins on line 3 of a Model 1 terminal and on line 2 of a Model 2 terminal. Ten lines are displayed including the heading line. The format of this display is the same as that available to the system operator when the DISPLAY spool queue command is entered. This display shows the status of the spool queues, not the data on the spool queues.

The user can enter the following characters starting in line 1 position 1 to alter the display:

Terminal Operator Keys	Resulting Display
F or ⍉	Pressing the F character key and the ENTER key, or the ENTER key alone causes the display to page forward if there are additional steps on the queue. Ten new lines are displayed including the heading. The display can be paged forward until the end of the queue is reached. A response at this time restarts the display at the beginning of the queue. If the job step displayed on the last line of the current display is altered through a CHANGE command or its position on the queue is changed while it is being displayed, then the display is restarted at the beginning of the queue even though a forward response was keyed. This is done so that the queue pointers can be reestablished by the spool file copy program.
Fnn	The F character along with a decimal number causes the display of the queue to be paged forward the number of positions indicated by nn. The decimal number can be a value of 1 through 99. If paging forward exceeds the end of the queue, the END OF QUEUE message is displayed and a forward response at this time restarts the display from the beginning of the queue. The queue display can also be restarted from the beginning as previously described (under F or ⍉) if the job step on the last line of the display is altered during the display.
C	This character causes the display function to be terminated.
WQ	These two characters cause the reader queue status to be displayed.
RQ	These two characters cause the reader queue status to be displayed.
PQ	These two characters cause the punch queue status to be displayed.
END	These three characters cause the spool file copy program to terminate.

CCP Assignment Set

The following statements could be provided in the CCP assignment set. The exact statements that you use depends on how you implement the spool file copy program under CCP.

// TERMATTR[1]	ATTRID-04,BLKL-nnn, DATAFORM-MESSAGE
// SYMFILE[2]	NAME-CONTROL,DISKFILE-'names of files to be used for CONTROL file; they must be defined in FILE statements during CCP STARTUP'
// SYMFILE[2]	NAME-READERQ,DISKFILE-'names of files to be used for READERQ file; they must be defined in FILE statements during CCP STARTUP'
// SYMFILE[2]	NAME-PRINTQ,DISKFILE-'names of files to be used for PRINTQ file; they must be defined in FILE statements during CCP STARTUP'
// SYMFILE[2]	NAME-PUNCHQ,DISKFILE-'names of files to be used for PUNCHQ file; they must be defined in FILE statements during CCP STARTUP'
// SYMFILE[2]	NAME-DISPLAYQ,DISKFILE-'names of files to be used for DISPLAYQ file; they must be defined in FILE statements during CCP STARTUP'

[1] The TERMATTR BLKL parameter must be a minimum of 817. The TERMATTR ATTRID parameter must be 04 (ATTRID-04) and be non-DFF. The TERMATTR DATAFORM parameter must be MESSAGE (DATAFORM-MESSAGE).

[2] The SYMFILE statements are not required if the PROGRAM statement lists all file names that will be referenced by the spool file copy program. The specific file to be used can then be specified on the FILE parameter of the spool file copy program control statement. If SYMFILE statements are not used and consecutive output files are listed in the PROGRAM statement, then only one copy of the spool file copy program can be loaded under CCP at a time. A FILE TEMPORARILY UNAVAILABLE message is issued if additional program requests for QCOPY are encountered while QCOPY is executing under CCP.

Note: If the spool file copy program is to be loaded from a pack other than the system pack, you must first use the copy function of the library maintenance program ($MAINT) to copy all modules starting with $QC (NAME-$QC.ALL) to the object library of that pack. The data management modules included by the spool file copy program must be located on either the program pack or the system pack.

// SYMFILE[1]	NAME-RESTORE,DISKFILE-'names of files to be used for RESTORE file; they must be defined in FILE statements during CCP STARTUP'
// PROGRAM[2]	NAME-QCOPY,PGMDATA-YES, FILES-'CONTROL/CG,[4] READERQ/CG,[5] PRINTQ/CO,[3] PUNCHQ/CO,[3] DISPLAYQ/CO,[3] RESTORE/CG,AUTHORIZ/DG[1] PACK-SYSTEM,TASKSIZE-12[6]

How to Request $QCOPY From a Terminal

To execute the spool file copy program under CCP for the preceding assignment set statements (if symbolic files are used), the terminal operator must enter the following statements:

/FILE	CONTROL,name of the file given in the CONTROL SYMFILE statement
/FILE	READERQ,name of the file given in the READERQ SYMFILE statement
/FILE	PRINTQ,name of the file given in the PRINTQ SYMFILE statement
/FILE	PUNCHQ,name of the file given in the PUNCHQ SYMFILE statement
/FILE	DISPLAYQ,name of the file given in the DISPLAYQ SYMFILE statement
/FILE	RESTORE,name of the file given in the RESTORE SYMFILE statement
QCOPY	

[1] The SYMFILE statements are not required if the PROGRAM statement lists all file names that will be referenced by the spool file copy program. The specific file to be used can then be specified on the FILE parameter of the spool file copy program control statement. If SYMFILE statements are not used and consecutive output files are listed in the PROGRAM statement, then only one copy of the spool file copy program can be loaded under CCP at a time. A FILE TEMPORARILY UNAVAILABLE message is issued if additional program requests for QCOPY are encountered while QCOPY is executing under CCP.

[2] CCP does not allow a program request of a $xxxxx program from a terminal. To execute under CCP, the spool file copy program must be renamed to a name (QCOPY for example) that does not begin with a dollar sign ($).

[3] The files accessed as CO (consecutive output) can also be accessed as CA (consecutive add). This allows the adding of records to the logical end of the file as well as sharing of the file after it is closed.

[4] CONTROL/CG can also be CONTROL/DG.

[5] READERQ/CG can also be READERQ/CO, READERQ/DG, READERQ/IS, or READERQ/ISL.

[6] The TASKSIZE parameter must be specified when the function requires more than 10K bytes.

Note: If the spool file copy program is to be loaded from a pack other than the system pack, you must first use the copy function of the library maintenance program ($MAINT) to copy all modules starting with $QC (NAME-$QC.ALL) to the object library of that pack. The data management modules included by the spool file copy program must be located on either the program pack or the system pack.

Examples

The following example shows the statements that the terminal operator keys to place the job named FIRST on the spool reader queue (the symbolic files were previously given in the assignment set).

Terminal Operator Keys	Message Returned
/FILE CONTROL,CONTROL1	ACCEPTED PROCEED
/FILE READERQ,RDRFL1	ACCEPTED PROCEED
/FILE PRINTQ,PRTFL1	ACCEPTED PROCEED
/FILE PUNCHQ,PCHFL1	ACCEPTED PROCEED
/FILE DISPLAYQ,DSYFL1	ACCEPTED PROCEED
/FILE RESTORE,REST1	ACCEPTED PROCEED
QCOPY ,pswd	UQ SP CS ENTER CONTROL STATEMENT
// COPYRDRQ UNIT-D2, RECL-96, INPUT-TERMINAL	UQ SP RQ ENTER RDRQ DATA
//FIRST JOB PARTITION-D, CORE-12	UQ SP RQ ENTER RDRQ DATA
// LOAD $MAINT,F1	UQ SP RQ ENTER RDRQ DATA
// RUN	UQ SP RQ ENTER RDRQ DATA
// COPY FROM-F1,TO-PRINT, LIBRARY-ALL,NAME-DIR	UQ SP RQ ENTER RDRQ DATA
// END	UQ SP RQ ENTER RDRQ DATA
/.	UQ SP RQ ENTER RDRQ DATA
/.	UQ SP CS ENTER CONTROL STATEMENT

// END if additional $QCOPY functions are required, they can be given here before you specify END.

Multiple copies of the spool file copy program can execute concurrently under CCP and/or in batch partitions.

Considerations for Terminating the Spool File Copy Program under CCP

The various combinations of control statement conditions and the operating mode determine whether the spool file copy program will terminate when its functions are complete or will read additional control statements from the requesting terminal. The following rules apply to these conditions and operating modes:

- The spool file copy program terminates whenever an END statement is read.
- If end of file is received when the spool file copy program is reading from a CONTROL file, the program terminates unless the reading of the file was initiated by a COPYCTRL statement read from the terminal.
- When a control statement error is detected, the next control statement is read from the requesting terminal; the spool file copy program terminates if the control statement was not requested from a 3275 or a 3277 terminal.

The following table provides a summary:

Control statement conditions	Operating mode			
		Reading control statements from a file		
			$QCOPY has a 3275 or a 3277 terminal	
	Reading control statements from the requesting terminal	$QCOPY does not have a 3275 or a 3277 terminal	Reading CONTROL file via program request	Reading CONTROL file via COPYCTRL
END statement	A	A	A	A
EOF received while reading control file	X	A	A	B
Control statement error	B	A	B	B

Legend:

A—Go to EJ
B—Read next control statement from the requesting terminal
X—Invalid condition

Recover Index Program—$RINDX

PROGRAM DESCRIPTION

The recover index program recovers added records to an indexed file.

The recover index program is used to recover the records added to an indexed file if, for any reason, the program adding the records is terminated before end of job.

The recover index program should be executed as soon as possible after the abnormal termination.

The functions of the recover index program for each file organization are:

- Indexed file
  - If added keys exist for the file when the abnormal termination occurs, $RINDX updates the end-of-index and end-of-data pointers. File information—defined as file label, file type, pack label, and file date—is printed. The last added key for this file is also printed.
  - If keys has not been added when the abnormal termination occurred, only the file information is printed.

- Consecutive file[1]
  - If a *consecutive* file is detected, only the file information is printed.

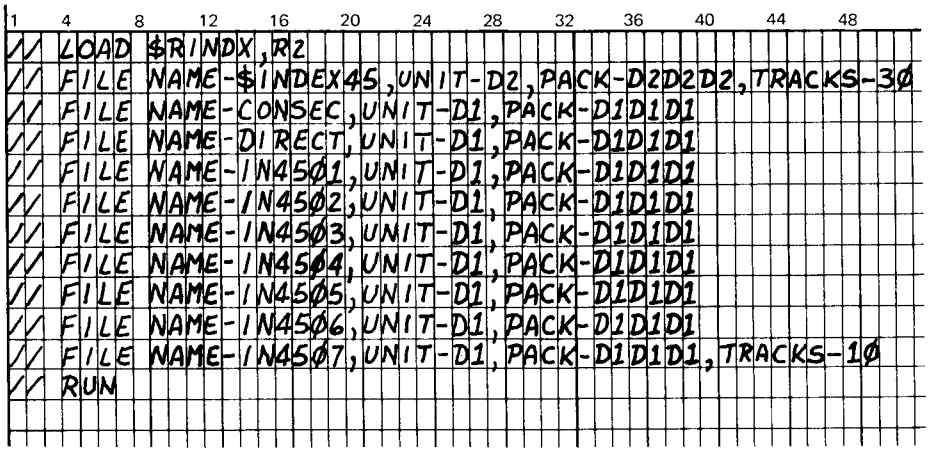
- Direct file[1]
  - If a *direct* file is detected, only the file information is printed.

- If the recover index program cannot find the file identified by the OCL FILE statement, the file information and the FILE NOT AVAILABLE message is printed.

[1] You can use the file recovery from physical address function of the copy/dump program ($COPY) to recover consecutive or direct files if an abnormal termination occurs when the system is adding to a consecutive file or creating a direct file.

FILE IDENTIFICATION

Each indexed file for which records are to be recovered must be identified by an OCL FILE statement. The only description that you must include is the file name, unit code, and pack ID. You may also include OCL FILE statements for other than indexed files; however, the recover index program does not attempt to recover records in other file organizations. The following example shows a FILE statement for each file to be checked for record recovery:



The $INDEX45 file (NAME-$INDEX45) and the $INDEX40 file (NAME-$INDEX40) are work files used to decrease the processing time for sorting the indexes of large indexed files. If you do not supply a $INDEX45 or a $INDEX40 FILE statement, the system attempts to allocate the work space. For additional information regarding the $INDEX45 and the $INDEX40 work files, see *File Facilities* in Part 2.

The following printout is a result of processing each FILE statement shown in the previous example:

```
$RINDX - 15/D FILE RECOVERY PROGRAM  DATE - XX/XX/XX   TIME-00.00.25

FILE      FILE PACK      FILE     LAST ADD KEY
LABEL     TYPE LABEL     DATE       INCLUDED

CONSEC     C   D1D1D1    021976
DIRECT     D   D1D1D1    021976
IN4501     I   D1D1D1    021976   00971
IN4502     I   D1D1D1    021976   002031
IN4503     I   D1D1D1    021976   0004131
IN4504     I   D1D1D1    021976   00005191
IN4505     I   D1D1D1    021976   000007211
IN4506     I   D1D1D1    021976   0000007211
IN4507         D1D1D1    021676   FILE NOT AVAILABLE

ALL FILES PROCESSED
```

The CI UL xx message occurs when all of the following conditions are present:

- The system is processing a FILE statement.
- The described file cannot be found.
- A TRACKS or RECORDS parameter is not included on the FILE statement.

If the recover index program cannot find the file identified by the FILE statement, the file information and the FILE NOT AVAILABLE message is printed.

After all OCL FILE statements have been processed, an ALL FILES PROCESSED message is printed. The index is then sorted and the VTOC (volume table of contents) updated.

Note: After the ALL FILES PROCESSED message is printed, do not cancel $RINDX or start the next job prior to actual end of job. Processing continues with sorting the index and updating the VTOC.

OCL CONSIDERATIONS

The following OCL statements are needed to load and execute the recover index program:

```
// LOAD $RINDX,code
// SWITCH 1xxxxxxx (this statement is optional)
// FILE NAME-xxxxxxxx,UNIT-xx,PACK-xxxxxx
// RUN
```

The code you supply depends on the location of the simulation area containing the recover index program. Possible codes are R1, F1, R2, F2.

CONSIDERATIONS AND RESTRICTIONS

- If more than one file has the same name, use DATE or LOCATION to further identify the file you intend to check.
- If a disk I/O error occurs during the execution of $RINDX, the file information and error message DISK I/O ERROR is printed. A system message then occurs; options are:
  - Continue processing with the next file
  - Cancel the job
- If message DDØP (Keysort Duplicate Key) occurs during the execution of $RINDX, it may indicate that the program was abnormally terminated during the process of sorting the index. Continue processing until end of job for $RINDX. If the file is not known to have duplicate keys, use the copy/dump program ($COPY) with REORG-NO and an OMIT or DELETE parameter to rebuild the index.
- Recovery of the last record of a file that spans a disk sector beyond the sector that contains the key is optional. If you choose to recover this record, you must include a SWITCH statement in the OCL. Otherwise, the program shifts the last record pointers to exclude this key and record, thereby ensuring that only valid records are recovered.

  The SWITCH statement must cause external indicator 1 to be on. External indicators 2 through 8 can be either on or off. For example:

  ```
  // SWITCH 1xxxxxxx
  ```

  When you include this statement and the last record meets the previously described conditions, the record may be invalid because the program cannot determine whether or not it is complete.

  For maximum recovery, the keys in an indexed file should be at the end of each record.
- If an abnormal termination occurs when a key(s) was added to the file but not written to the disk, the end-of-index pointer will not include this key(s) after the system has executed the recovery index program.

EXAMPLES

In the following example, the recover index program is loaded from R2. The printout shows that keys were added to each of the files except IN4403 before the abnormal termination.

```
// LOG PRINTER
// LOAD $RINDX,R2
// FILE NAME-$INDEX45,UNIT-D2,PACK-D2D2D2,TRACKS-30
// FILE NAME-IN4501,UNIT-D1,PACK-D1D1D1
// FILE NAME-IN4502,UNIT-D1,PACK-D1D1D1
// FILE NAME-IN4503,UNIT-D1,PACK-D1D1D1
// FILE NAME-IN4401,UNIT-D1,PACK-D1D1D1
// FILE NAME-IN4402,UNIT-D1,PACK-D1D1D1
// FILE NAME-IN4403,UNIT-D1,PACK-D1D1D1
// RUN


$RINDX - 15/D FILE RECOVERY PROGRAM  DATE - XX/XX/XX  TIME-00.00.25

FILE      FILE PACK      FILE     LAST ADD KEY
LABEL     TYPE LABEL     DATE       INCLUDED

IN4401     I   D1D1D1    022676  00000000000000000000971
IN4402     I   D1D1D1    022676  00000000000000000002031
IN4403     I   D1D1D1    022676
IN4501     I   D1D1D1    022676  00971
IN4502     I   D1D1D1    022676  002031
IN4503     I   D1D1D1    022676  0004131

ALL FILES PROCESSED
  1 DD KS     I                $RINDX01
BEGIN KEY SORT/MERGE - IN4401
  1 DD KS     I                $RINDX01
BEGIN KEY SORT/MERGE - IN4402
  1 DD KS     I                $RINDX01
BEGIN KEY SORT/MERGE - IN4403
  1 DD KS     I                $RINDX01
BEGIN KEY SORT/MERGE - IN4501
  1 DD KS     I                $RINDX01
BEGIN KEY SORT/MERGE - IN4502
  1 DD KS     I                $RINDX01
BEGIN KEY SORT/MERGE - IN4503




 1 CT EJ      I                $RINDX01
02/26/76 00.00.19 00.02.47
```

Reassign Alternate Track Program—$RSALT

PROGRAM DESCRIPTION

The reassign alternate track program has the following function:

Reassign alternate tracks on a 3340 data module for use on System/360 or System/370.

When it is necessary to transport a 3340 data module from System/3 to System/360 or System/370, you must execute the reassign alternate track program ($RSALT) before you execute the DOS/OS initialization program.

On a 3340 data module initialized on System/3, there are 40 alternate tracks on cylinders 167 and 168. On a System/360 or System/370 3340 data module, there are 24 alternate tracks on cylinders 208 and 209. Consequently, if a 3340 data module initialized on System/3 has more than 24 defective primary tracks, it cannot be initialized by System/360 or System/370.

Note: Data interchange is not supported between the System/3 and the System/360 or System/370, so this program cannot be used for that purpose. System/3 data existing on the data module before $RSALT is run will be lost.

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

Function	Control Statements
Reassign Alternate Tracks	// ALTA UNIT-{code / 'codes' / ALL},PACK-{name / 'names'} // END

PARAMETER SUMMARY

Parameter	Description
UNIT-{code / 'codes' / ALL}	Specifies the location of the 3340 data module. Possible codes are D1, D2, D3 or D31, D4 or D41, and ALL.
PACK-{name / 'names'}	Specifies the name of the data module you want to modify.

PARAMETER DESCRIPTIONS

UNIT Parameter

The UNIT parameter (UNIT-{code / 'codes' / ALL}) specifies the location of the 3340 data module you want to modify. $RSALT can modify a maximum of three 3340 data modules during one execution of the program. The code you supply depends on the number of data modules that you want to modify. For example:

For one data module, specify UNIT-code
For two data modules, specify UNIT-'code,code'
For three data modules, specify UNIT-'code,code,code'

If UNIT-ALL is specified, $RSALT modifies all data modules that are online and not being used by other partitions.

Code	Meaning
D1	Modify data module on drive 1
D2	Modify data module on drive 2
D3 or D31	Modify data module on drive 3
D4 or D41	Modify data module on drive 4
ALL	Modify all available data modules

If the IPL was performed from drive 1, $RSALT will not accept D1 as a valid code.

PACK Parameter

The PACK parameter (PACK-{name / 'names'}) tells the program the name of the data module to be modified. The name must not exceed 6 characters and can be any of the standard System/3 characters except apostrophes, commas, and leading or embedded blanks.

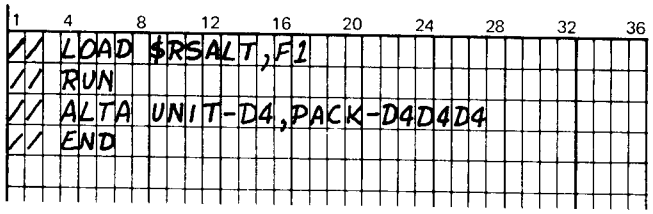
When more than one data module is modified, the associated pack name and data module code must be in the same relative position within each parameter. For example, in the following statement the name D2D2D2 is associated with unit code D2. Also, the name D3D3D3 is associated with unit code D3:

```
// ALTA UNIT-'D2,D3',PACK-'D2D2D2,D3D3D3'
```

If UNIT-ALL is specified, PACK parameter names must be supplied for all data modules that are to be modified.

The reassign alternate track program compares the name in the PACK parameter with the name on the data module to ensure that they match. If not, the program halts with an error message.

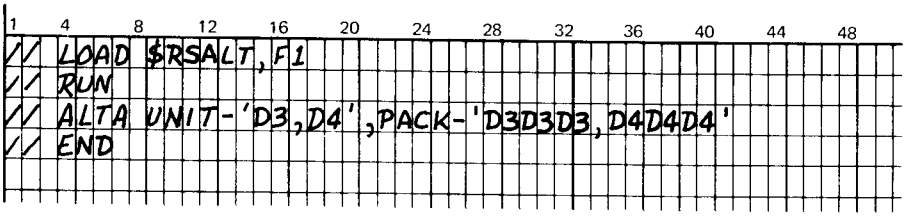
Figures 4-103, 4-104, and 4-105 are examples of OCL and control statements required to execute the $RSALT program.



Explanation:

- The reassign alternate track program is loaded from F1.
- The data module mounted on drive 4 (UNIT-D4) is modified to System/360-System/370 format.
- The name of the data module is compared with D4D4D4 (PACK-D4D4D4).

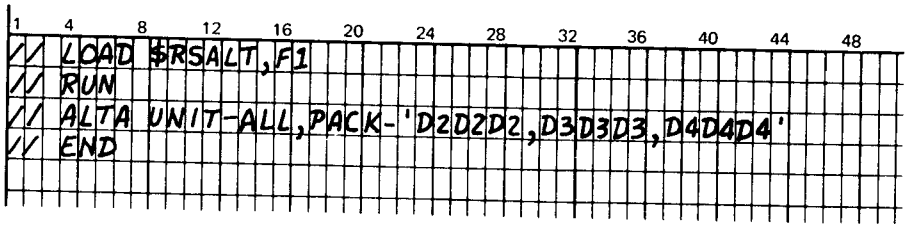
Figure 4-103. Modify One Data Module



Explanation:

- The reassign alternate track program is loaded from F1.
- The data modules mounted on drive 3 and drive 4 (UNIT-'D3,D4') are modified to System/360-System/370 format.
- The name of each data module is compared with its associated PACK parameter name (PACK-'D3D3D3, D4D4D4').

Figure 4-104. Modify Two Data Modules



Explanation:

- The reassign alternate track program is loaded from F1.
- The data modules on drives 2, 3, and 4 (UNIT-ALL) are to be modified to System/360-System/370 format.
- The name on each data module is compared with its associated PACK parameter name (PACK-'D2D2D2, D3D3D3,D4D4D4').

Figure 4-105. Modify Multiple Data Modules

Simulation Area Program—$SCOPY

PROGRAM DESCRIPTION

The simulation area program has the following functions:

- COPYAREA copies the entire contents of one simulation area to another simulation area.
- CLEAR clears all the data from a simulation area and builds a simulated cylinder 0 (optionally provides volume ID and owner ID).
- NEWNAME changes the name (volume ID) and/or owner ID of a simulation area.
- NAMES prints and/or displays the name (volume ID) of each available simulation area and main data area.
- MOVE copies the entire contents of one simulation area to another simulation area, clears the simulation area from which the contents were copied, and builds a simulated cylinder 0 in the simulation area copied from.
- COPYIPL copies IPL records from one volume to another volume.

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

Functions	Control Statements
COPYAREA	// COPYAREA FROM-code,TO-code,AREA-name[,PACK-name] [,TONAME-name] [,SYSTEM-{YES / NO}]
CLEAR	// CLEAR FROM-code[,PACK-name] [,AREA-name] [,CLRNAME-name] [,ID-name] [,TYPE-{CHECK / FORCE}]
NEWNAME	// NEWNAME TO-code,AREA-name,TONAME-name[,PACK-name] [,ID-name]
NAMES	// NAMES [PRINT / DISPLAY / BOTH]
MOVE	// MOVE FROM-code,TO-code,AREA-name[,PACK-name] [,TONAME-name] [,ID-name] [,SYSTEM-{YES / NO}] [,CLRNAME-name]
COPYIPL	// COPYIPL FROM-code,TO-code,PACK-name // END

Figure 4-106 lists the possible codes that can be supplied in each of the appropriate parameters for $SCOPY. Before using these codes, you should be familiar with the format of each volume and the assignment of the codes as described in Part 3 of this manual.

Disk Drive Type and Designation	Main Data Area Codes	Simulation Area Codes[1]
3340 drive 1	D1	D1A, D1B, D1C, D1D
3340 drive 2	D2	D2A, D2B, D2C, D2D
3340 drive 3	D3 or D31	D3E, D3A
3340 drive 4	D4 or D41	D4E, D4A
3344 drive 3		
volume 1	D3 or D31	D3E, D3A
volume 2	D32	D3F, D3B
volume 3	D33	D3G, D3C
volume 4	D34	D3H, D3D
3344 drive 4		
volume 1	D4 or D41	D4E, D4A
volume 2	D42	D4F, D4B
volume 3	D43	D4G, D4C
volume 4	D44	D4H, D4D

[1] Instead of a simulation area code, a 5444 unit code (F1, R1, F2, R2) may be used, and when used, it refers to the simulation area currently assigned to that code in the partition in which $COPY is being executed.

Figure 4-106. Main Data Area and Simulation Area Codes

PARAMETER SUMMARY

Parameter	Description
COPYAREA	
FROM-code	Identifies the simulation area being copied. See Figure 4-106 for the possible codes.
TO-code	Identifies the simulation area receiving the copy. See Figure 4-106 for the possible codes.
PACK-name	Identifies the name of the main data area associated with the simulation area receiving the copy.
AREA-name	Identifies the name of the simulation area being copied.
TONAME-name	Specifies a name change for the simulation area receiving the copy.
SYSTEM-{YES / NO}	Specifies whether IPL information is to be copied.
CLEAR	
FROM-code	Identifies the simulation area being cleared. See Figure 4-106 for the possible codes.
PACK-name	Specifies the name of the main data area associated with the simulation area to be cleared.
AREA-name	Specifies the simulation area to be cleared. This parameter cannot be specified if the simulation area does not have an assigned name. PID001 must be specified to clear an area used for distribution of programs from the IBM program library/PID.
CLRNAME-name	Specifies the name to be given to the simulation area being cleared. This parameter must be specified if the simulation area does not have an assigned name. The name of the simulation area, if previously defined, will remain the same if this parameter is not specified.
ID-name	Enables you to use a 10-character name in addition to the simulation area name to further identify a simulation area.
TYPE-CHECK	Tells the program to check for active files or libraries and halt if any are found before clearning the simulation area.
TYPE-FORCE	Tells the program to clear the simulation area without checking for active files or libraries.

Parameter	Description
NEWNAME	
TO-code	Specifies the name of the simulation area being renamed. See Figure 4-106 for the possible codes.
PACK-name	Specifies the name of the main data area associated with the simulation area being renamed.
AREA-name	Specifies the existing name of the simulation area being renamed.
TONAME-name	Specifies the new name being given to the simulation area. The name may be up to 6 characters in length and contain any combination of standard System/3 characters except apostrophes, embedded blanks, and embedded commas (due to their delimiter function). See Appendix A for a list of standard System/3 characters.
ID-name	Enables you to assign a 10-character name in addition to the area name to further identify the simulation area.
NAMES	
PRINT DISPLAY BOTH	The PRINT parameter indicates that the names of all online simulation areas are to be printed on the device assigned as the system print device (PRINTER statement). The DISPLAY parameter indicates that the names of all online simulation areas are to be displayed on the display screen. The BOTH parameter indicates that the names of all online simulation areas are to be printed and displayed. If no parameter is specified, the simulation area names are printed. If an area is unavailable, its volume ID is left blank and a message will indicate the reason. Using DISPLAY, an additional screen is used to display exception output.
MOVE	
FROM-code	Identifies the simulation area being moved. See Figure 4-106 for the possible codes.
TO-code	Identifies the simulation area receiving moved information. See Figure 4-106 for the possible codes.
PACK-name	Identifies the name of the main data area associated with the simulation area receiving the moved information.
AREA-name	Specifies the name of the simulation area being moved.
TONAME-name	Specifies a name change for the simulation area receiving the moved information.

Parameter	Description
ID-name	Name can be a maximum of 10 characters and is used to further identify the simulation area being moved.
SYSTEM-{YES / NO}	Specifies whether IPL information is to be copied.
CLRNAME-name	Used to assign a name to the area from which the information has been moved.
COPYIPL	
FROM-code	Identifies the main data area containing the IPL records to be copied. See Figure 4-106 for the possible codes.
TO-code	Identifies the main data area receiving the IPL records. See Figure 4-106 for the possible codes.
PACK-name	Identifies the name of the main data area receiving the IPL records.

PARAMETER DESCRIPTIONS—COPYAREA

FROM and TO Parameters

The FROM parameter (FROM-code) identifies the simulation area that contains the information to be copied. The TO parameter (TO-code) identifies the simulation area that is to receive the copy. Possible codes are those for the simulation areas and the 5444 unit codes. See Figure 4-106 for possible codes.

PACK Parameter

The PACK parameter (PACK-name) identifies the name of the main data area associated with the simulation area receiving the copy. This is the name assigned by the disk initialization program ($INIT).

AREA Parameter

The AREA parameter (AREA-name) identifies the name of the simulation area that is to be copied.

Note: Using a COPYAREA or MOVE statement, the receiving simulation area is assigned the owner ID of the simulation area being copied from.

TONAME Parameter

The TONAME parameter (TONAME-name) is used to change the name of the simulation area that is to receive the copy. The name may contain up to 6 characters (see *CLRNAME Parameter* under *CLEAR* for explanation of valid names). If the TONAME parameter is omitted, the name of the simulation area that is to be copied is used.

SYSTEM Parameter

The SYSTEM parameter is used to copy IPL information. If SYSTEM-YES is specified, the IPL information from cylinder 0 of the volume from which an IPL was performed (D1, D3, or D31) is copied to cylinder 0 of the volume receiving the copied information. If SYSTEM-NO is specified, the IPL information is not copied. If this parameter is not specified, SYSTEM-NO is assumed.

PARAMETER DESCRIPTIONS—CLEAR

FROM Parameter

The FROM parameter (FROM-code) identifies the simulation area to be cleared. Codes that may be used are those for the simulation areas and the 5444 unit codes. See Figure 4-106 for possible codes.

PACK Parameter

The PACK parameter (PACK-name) specifies the name of the main data area associated with the simulation area that is to be cleared. This is the name assigned by the disk initialization program ($INIT).

AREA Parameter

The AREA parameter (AREA-name) specifies the name of the simulation area that is to be cleared. This parameter cannot be specified if the area has no assigned name. The AREA parameter must be specified as PID001 in order to clear an area used for distribution of programs from the IBM program library.

CLRNAME Parameter

The CLRNAME parameter (CLRNAME-name) specifies the name to be given to the simulation area that is to be cleared. The name may be up to 6 characters in length and contain any combination of standard System/3 characters except apostrophes, embedded blanks, and embedded commas (due to their delimiter function). See Appendix A for a list of standard System/3 characters. Some examples of valid area names are: 0, F0001, 012, A1B9, and ABC. If this parameter is not specified, the name of the simulation area is the name previously defined. If a name has not been previously defined, CLRNAME must be specified.

ID Parameter

The ID parameter enables you to include a maximum of 10 characters in addition to the area name to further identify a volume. The characters can be any combination of standard System/3 characters (see Appendix A) except apostrophes, blanks, and commas (due to their delimiter function). The information is strictly for volume identification; it is not used by the system for checking purposes. If no parameter is specified, the owner ID area in the volume label is blank.

TYPE Parameter

The TYPE parameter specifies the type of clear that is to be done. If TYPE-CHECK is specified, a check is made for active files or libraries. If any are found, the system issues an error message. If TYPE-FORCE is specified, the area is cleared without a check for active files or libraries. All libraries and data files are deleted. (Default is TYPE-CHECK.)

PARAMETER DESCRIPTIONS—NEWNAME

TO Parameter

The TO parameter (TO-code) identifies the simulation area that is to be renamed. Possible codes are those for the simulation areas and the 5444 unit codes. See Figure 4-106 for the possible codes.

PACK Parameter

The PACK parameter (PACK-name) specifies the name of the main data area associated with the simulation area being renamed. This is the name assigned by the disk initialization program ($INIT).

AREA Parameter

The AREA parameter (AREA-name) specifies the existing name of the simulation area that is to be renamed.

ID Parameter

The ID parameter (ID-name) specifies the owner ID that is to be given to the simulation area from which information was moved. If no parameter is specified, the owner ID in the volume label is left blank.

Note: Using a COPYAREA or MOVE statement, the receiving area is assigned the owner ID of the area being copied from. The owner ID name may be up to 10 characters in length. See *ID Parameter* under *CLEAR* for an explanation of valid names.

SYSTEM Parameter

The SYSTEM parameter is used to copy IPL information. If SYSTEM-YES is specified, the IPL information from cylinder 0 of the volume from which the IPL was performed (D1, D3 or D31) is copied to cylinder 0 of the volume receiving the copied information. If SYSTEM-NO is specified, the IPL information is not copied. If this parameter is not specified, SYSTEM-NO is assumed.

CLRNAME Parameter

The CLRNAME parameter (CLRNAME-name) is used to assign a name to the simulation area from which the information has been moved. The name may be up to 6 characters. See *CLRNAME Parameter* under *CLEAR* for an explanation of valid names. If no parameter is specified, the area is cleared and the name previously assigned is used.

PARAMETER DESCRIPTIONS—COPYIPL

FROM and TO Parameters

The FROM parameter (FROM-main data area code) identifies the volume containing the IPL records that are to be copied. The TO parameter (TO-main data area code) identifies the volume that is to receive the IPL records. See Figure 4-106 for the possible codes.

Note: COPYIPL cannot be executed to the volume from which an IPL was performed. For example, if an IPL was performed from DISK 1 (F1 or R1), you cannot specify D1 on the TO parameter. Also, if an IPL was performed from DISK 3 (F1 or R1), you cannot specify D3 or D31 on the TO parameter.

PACK Parameter

The PACK parameter (PACK-name) identifies the name of the main data area that is to receive the IPL records (the name was assigned by the disk initialization program ($INIT)).

TONAME Parameter

The TONAME parameter (TONAME-name) specifies the new name to be given to the simulation area. The new name may be up to 6 characters in length. See *CLRNAME Parameter* under *CLEAR* for an explanation of valid names.

PARAMETER DESCRIPTIONS–NAMES

PRINT Parameter

The PRINT parameter indicates that the names of all online simulation areas are to be printed on the device assigned as the system print device (PRINTER statement). The DISPLAY parameter indicates that the names of all online simulation areas are to be displayed on the display screen. The BOTH parameter indicates that the names of all online simulation areas are to be printed and displayed. If no parameter is specified, the simulation area names are printed. If an area is unavailable, its volume ID is left blank and a message will indicate the reason. Using DISPLAY, an additional screen is used to display exception output.

PARAMETER DESCRIPTIONS–MOVE

FROM and TO Parameters

The FROM parameter (FROM-code) identifies the simulation area that is to be moved. The TO parameter (TO-code) identifies the simulation area that is to receive the moved information. Possible codes are those for the simulation areas and the 5444 unit codes. See Figure 4-106 for the possible codes.

PACK Parameter

The PACK parameter (PACK-name) identifies the name of the main data area associated with the simulation area that is to receive the moved information. The name was assigned by the disk initialization program ($INIT).

AREA Parameter

The AREA parameter (AREA-name) specifies the name of the simulation area to be moved.

TONAME Parameter

The TONAME parameter (TONAME-name) is used to change the name of the simulation area that is to receive the information. If no parameter is specified, the name of the simulation area that is to be moved is used. See *CLRNAME Parameter* under *CLEAR* for an explanation of valid names.

OCL CONSIDERATIONS

The following OCL statements are needed to load the simulation area program:

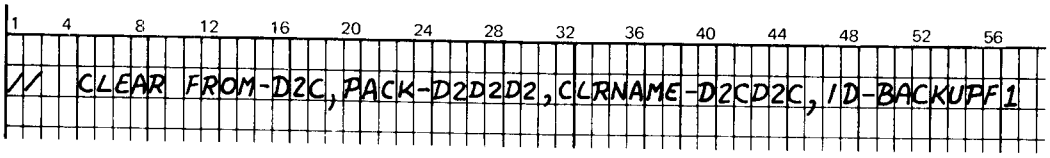
```
// LOAD $SCOPY,code
// RUN
```

The code you supply depends on the location of the simulation area containing the simulation area program. Possible codes are R1, F1, R2, and F2.

EXAMPLES

Figures 4-107 through 4-113 are examples of control statements used to perform specific functions of the simulation area program.

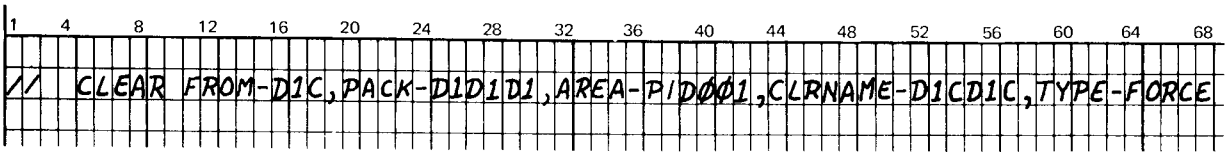
Figures 4-114 through 4-117 are examples of the NAMES display function.



Explanation:

After a check for active files and libraries (default is TYPE-CHECK), the simulation area (D2C) is cleared. It is given a volume ID of D2CD2C and an owner ID of BACKUPF1. This is an example of the CLEAR that is to be run after the entire volume has been initialized by $INIT.

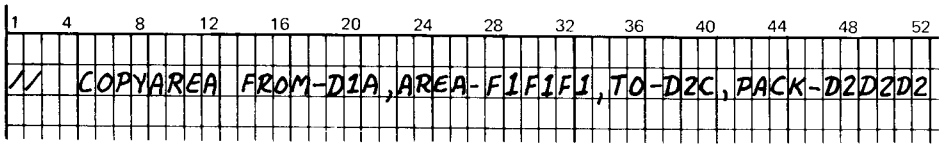
Figure 4-107. CLEAR Example: Clearing a Simulation Area



Explanation:

After verification that the volume ID on the simulation area (D1C) is PID001, the simulation area is cleared and given a volume ID of D1CD1C. The owner ID is all blanks and the check for active files and libraries is bypassed. This is an example of the control statement needed to clear a simulation area containing programs from the IBM program library/PID.

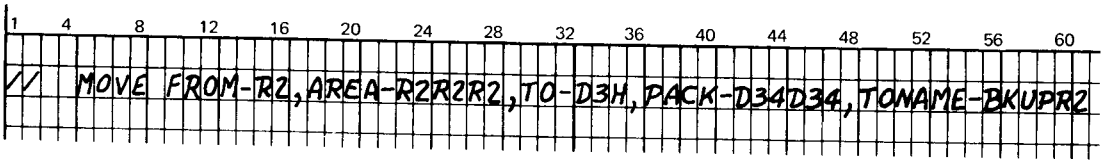
Figure 4-108. CLEAR Example: Clearing an Area Containing IBM Programs.



Explanation:

After verification that the volume ID of area D1A is F1F1F1, the simulation area (D1A) is copied to the simulation area (D2C). The entire simulation area is copied including cylinder 0, the volume ID, and the owner ID if it was present on D1A.

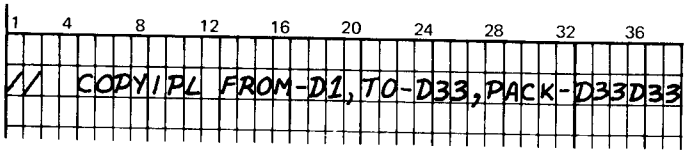
Figure 4-109. COPYAREA Example: Copy an Entire Simulation Area



Explanation:

The entire R2 simulation area (assigned to the partition in which $SCOPY is running) is copied to the simulation area (D3H) and is given a volume ID of BKUPR2 and an owner ID of the R2 area if one exists. After the copy is complete, the R2 simulation area is cleared of all data, its owner ID field is blank, and it retains its volume ID of R2R2R2. The R2 simulation area is now ready to be the receiving area of a COPYAREA or another MOVE.

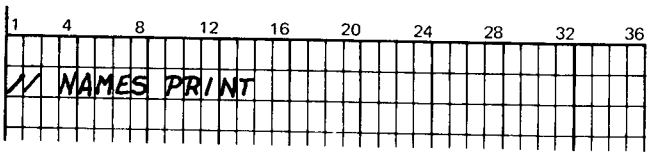
Figure 4-110. MOVE Example: Copy an Entire Simulation Area with New Volume ID



Explanation:

The IPL (initial program load) records and the 3340 microcode are copied from cylinder 0 of volume 1 to cylinder 0 of volume 33. A check is made before the copy to ensure that the volume ID of volume 33 is D33D33.

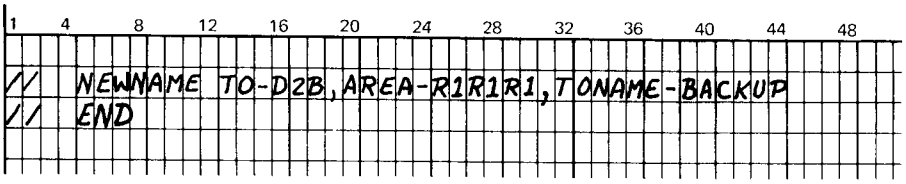
Figure 4-111. COPYIPL Example: Copy Cylinder 0 from Volume 1 to Volume 33



Explanation:

This control statement enables you to print on the system print device the volume ID of all online and available volumes. This control statement also provides the capability to print an exception line, if needed, giving the reason for any unavailable simulation area.

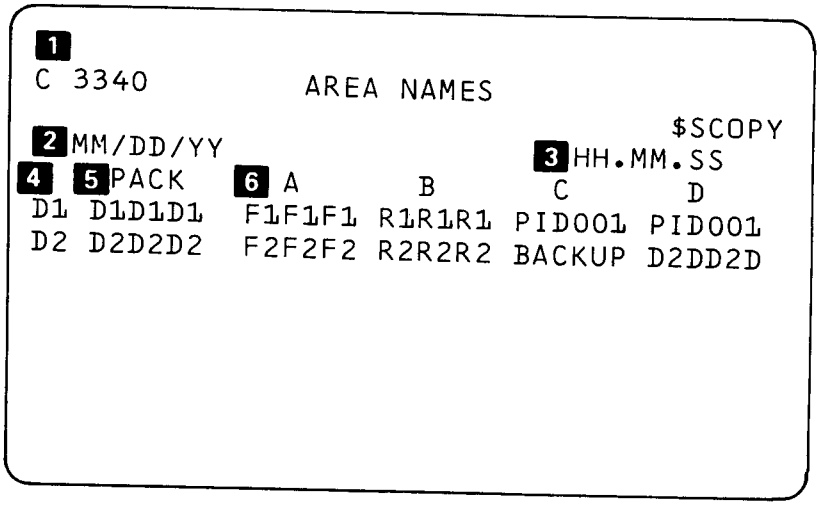
Figure 4-112. PRINT Example: Print ID Information



Explanation:

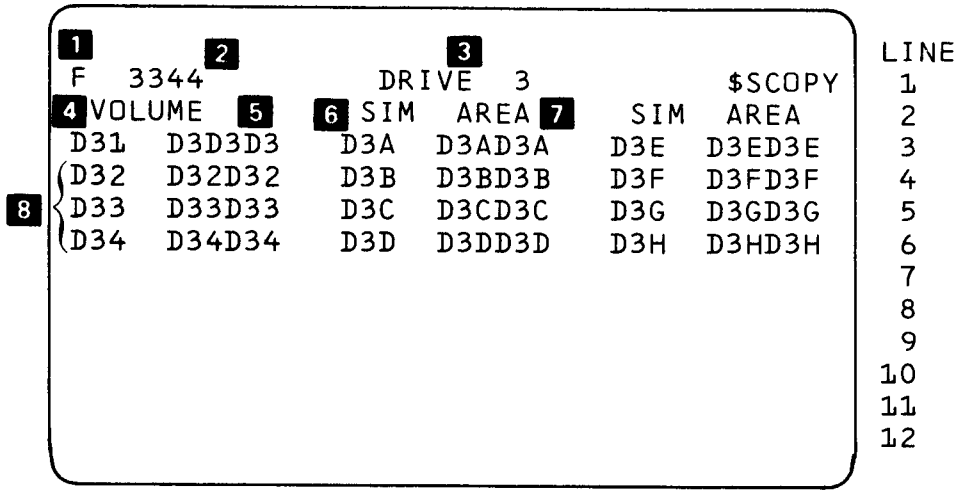
After verifying that the name (volume ID) of the simulation area (D2B) is R1R1R1, the program changes the name (volume ID) of the simulation area from R1R1R1 to BACKUP.

Figure 4-113. NEWNAME Example: Changing Volume ID



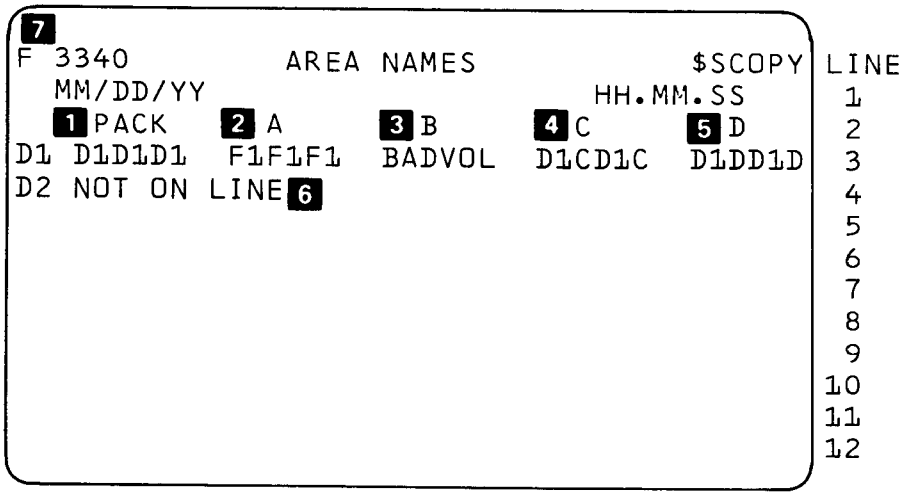
Item	Explanation
1	Operator response C – Cancel display and return to read the next control statement F – Page forward to next display B – Page backward to previous display The first display screen will have a C in this position if there is no subsequent display. It will have an F in this position to indicate that a subsequent display containing additional information is available. The last output page with available data will have a C in this position.
2	The system date
3	The system time (if full time support)
4	Main data area code
5	The ID of the volume
6	The volume ID of the simulation area

Figure 4-114. Example of Output from a NAMES Display



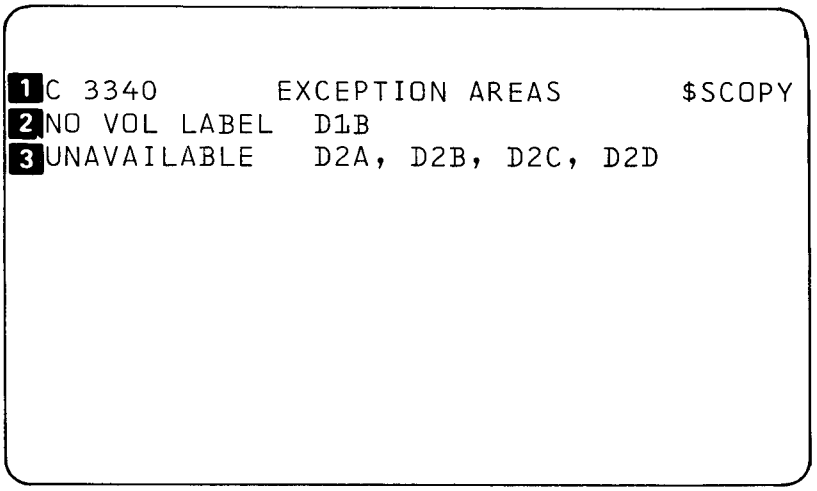
Item	Explanation
1	Operator response C – Cancel display and return to read the next control statement F – Page forward to next display B – Page backward to previous display The first display screen will have a C in this position if there is no subsequent display. It will have an F in this position to indicate that a subsequent display containing additional information is available. The last output page with available data will have a C in this position.
2	Type of disk drive
3	Disk drive number
4	Main data area code
5	The volume ID of the main data area
6	The simulation area code
7	The volume ID of the simulation area
8	These three lines are displayed only if drive 3 is a 3344

Figure 4-115. Example of Output from a NAMES Display



Item	Explanation
1	Drive D1 was available; therefore, the volume ID is displayed.
2	Simulation area D1A was available; therefore, its volume ID is displayed.
3	Simulation area D1B had no volume ID; therefore, BAD VOL is displayed.
4	Simulation area D1C was available; therefore, its volume ID is displayed.
5	Simulation area D1D was available; therefore, its volume ID is displayed.
6	Volume D2 is not ready; therefore, its volume ID and the volume IDs of the simulation areas are not displayed.
7	The F (forward code) appears indicating that a subsequent display containing additional information is available.

Figure 4-116. Example of NAMES Display with Exception Output



Item	Explanation
1	The C (cancel code) appears indicating that this is the last display page.
2	Simulation area D1B did not have a valid volume label.
3	The designated simulation areas were not available to $SCOPY.

Figure 4-117. Example of Exception Output with NAMES Display Statement

Tape Initialization Program—$TINIT

PROGRAM DESCRIPTION

The tape initialization program has the following functions:

- Check labeled tapes for a volume label and an unexpired file before writing a new volume label.
- Clear labeled or unlabeled tapes by bypassing CHECK and unconditionally initializing the tape.
- Display the volume and header labels.

The tape initialization program prepares magnetic tapes for use. This program writes IBM standard volume labels on tape in order for tape data management to perform IBM standard label processing.

All tapes must be initialized before use. Tapes that have been initialized need not be reinitialized unless you want to write a new volume label or use a tape that contains a permanent file for output. This program can either initialize (CLEAR or CHECK) or display (DISPLAY) one tape per unit during the same program run.

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

Functions	Control Statements

Check for an expired file and a label, then write a new label.	// VOL UNIT-{T1 / T2 / T3 / T4},REEL-{NL / xxxxxx},[TYPE-CHECK] [,ASCII-{YES / NO}]
	① [,DENSITY-{1600 / 800 / 556 / 200}] [,ID-xx...xx]
	// END

Write volume label without checking for old label.	// VOL UNIT-{T1 / T2 / T3 / T4},REEL-{NL / xxxxxx},TYPE-CLEAR [,ASCII-{YES / NO}]
	① [,DENSITY-{1600 / 800 / 556 / 200}] [,ID-xx...xx]
	// END

Display volume and header labels.	// VOL UNIT-{T1 / T2 / T3 / T4},TYPE-DISPLAY ① [,DENSITY-{800 / 556 / 200}] [,FILES-{001 / xxx / ALL}] // END

① If density is not specified, the default for 7-track tape units is 800 bpi; the default for 9-track tape units is 1600 bpi. Other defaults are underlined.

The DENSITY parameter on display volume label is valid only for 7-track tape units.

Valid density for 7-track tape units is 200, 556, and 800 bpi. Valid density for 9-track tape units is 800 bpi or 1600 bpi.

PARAMETER SUMMARY

Parameters	Description
TYPE-CHECK (Optional, select one) ①	Check to see if the file has expired; then write a new label. Do not use this on blank tapes because the program attempts to read a blank tape, causing tape runaway. On multifile volumes, only the first file is checked.
TYPE-CLEAR (Optional, select one) ①	Write a new volume label without checking for an expired file.
TYPE-DISPLAY (Optional, select one) ①	Print the contents of the volume label and the header labels.
UNIT-code (required)	Specifies which tape drive contains the tape to be initialized. Possible codes are: T1, T2, T3, and T4. A separate VOL statement is needed for each tape unit that contains a tape to be initialized, checked, or displayed.
REEL-NL (Required for TYPE-CHECK and TYPE-CLEAR)	Specifies that an unlabeled tape is to be prepared.
REEL-xxxxxx (Required for TYPE-CHECK and TYPE-CLEAR)	Specifies the volume serial number that the tape initialization program writes on tape. Must be A-Z, 0-9, $, #, or @.
ASCII-YES (optional)	The tape is written in ASCII code. This is invalid for 7-track tape.
ASCII-NO (optional)	The tape is written in EBCDIC code. If the ASCII parameter is omitted, NO is assumed.
DENSITY-200 (Optional, select one)	The tape is written at a density of 200 bpi (bits per inch). The file written on this tape unit must be written at this density.
DENSITY-556 (Optional, select one)	The tape is written at a density of 556 bpi (bits per inch). The file written on this tape unit must be written at this density.
DENSITY-800 (Optional, select one)	The tape is written at a density of 800 bpi (bits per inch). The file written on this tape unit must be written at this density.
DENSITY-1600 (Optional, select one)	The tape is written at a density of 1600 bpi (bits per inch). The file written on this tape unit must be written at this density.
ID-xxxxxxxxxx (Optional)	Provides an additional identification field. This field is not processed by the system. A maximum of 10 characters can be used if ASCII-NO is specified. If ASCII-YES is specified, 14 characters can be used.
FILES-001 (Optional)	Indicates the number of files for which header labels are to be displayed. If this parameter is omitted, 001 is assumed. The code nnn can be 1 to 999 (leading zeros are not required). If FILES-ALL is specified, all labels are printed.
FILES-nnn (Optional)	
FILES-ALL (Optional)	

① If TYPE is not specified, TYPE-CHECK is assumed.

OCL CONSIDERATIONS

The following OCL statements are needed to load the tape initialization program.

```
// LOAD $TINIT,code
// RUN
```

The code you supply depends on the location of the simulation area containing the program. Possible codes are R1, F1, R2, F2.

PRINTOUT OF VOLUME LABEL

The following sample jobs show the format of data printed by the tape initialization program from a 9-track tape unit and from a 7-track tape unit.

Note: The tape volume for each unit contains five files.

```
// LOAD $TINIT,F1
// RUN
// VOL TYPE=DISPLAY,FILES=ALL,UNIT=T1
// VOL TYPE=DISPLAY,FILES=ALL,UNIT=T2
// VOL TYPE=DISPLAY,FILES=ALL,UNIT=T3
// VOL TYPE=DISPLAY,FILES=ALL,UNIT=T4
// END
```

```
*****     TAPE LABEL DISPLAY          DATE-11/11/77          TIME-00.02.34     *****
          UNIT-T1          FILE SERIAL-TAPE1          OWNER CODE

----------HEADER 1----------                    ----------HEADER 2----------
FILE SEQ  FILE-IDENTIFIER   VOL SEQ  CREATE DATE  EXPIRE DATE  FILE SERIAL  REC FORM  BLK ATTR  REC LENG  BLOCK LENG  RECORDING TECHNIQUE  PRTR CNTRL  JOBNAME  STEPNAME
0001 FILE1                  0001     77315        77315        TAPE1        V         B         00100     00524                                  /DMTAPEC1
0002 FILE2                  0001     77315        77315        TAPE1        V         B         00020     00100                                  /DMTAPEC1
0003 FILE3                  0001     77315        77315        TAPE1        F         B         00030     00090                                  /DMTAPEC1
0004 FILE4                  0001     77315        77315        TAPE1        F         B         00040     00080                                  /DMTAPEC1
0005 FILE5                  0001     77315        77315        TAPE1        F         B         00050     00050                                  /DMTAPEC1

*****     TAPE LABEL DISPLAY          DATE-11/11/77          TIME-00.02.38     *****
          UNIT-T2          FILE SERIAL-TAPE2          OWNER CODE

----------HEADER 1----------                    ----------HEADER 2----------
FILE SEQ  FILE-IDENTIFIER   VOL SEQ  CREATE DATE  EXPIRE DATE  FILE SERIAL  REC FORM  BLK ATTR  REC LENG  BLOCK LENG  RECORDING TECHNIQUE  PRTR CNTRL  JOBNAME  STEPNAME
0001 FILE1                  0001     77315        77315        TAPE2        V         B         00100     00524                                  /DMTAPEC1
0002 FILE2                  0001     77315        77315        TAPE2        V         B         00020     00100                                  /DMTAPEC1
0003 FILE3                  0001     77315        77315        TAPE2        F         B         00030     00090                                  /DMTAPEC1
0004 FILE4                  0001     77315        77315        TAPE2        F         B         00040     00080                                  /DMTAPEC1
0005 FILE5                  0001     77315        77315        TAPE2        F         B         00050     00050                                  /DMTAPEC1

*****     TAPE LABEL DISPLAY          DATE-11/11/77          TIME-00.02.42     *****
          UNIT-T3          FILE SERIAL-TAPE3          OWNER CODE

----------HEADER 1----------                    ----------HEADER 2----------
FILE SEQ  FILE-IDENTIFIER   VOL SEQ  CREATE DATE  EXPIRE DATE  FILE SERIAL  REC FORM  BLK ATTR  REC LENG  BLOCK LENG  RECORDING TECHNIQUE  PRTR CNTRL  JOBNAME  STEPNAME
0001 FILE1                  0001     77315        77315        TAPE3        V         B         00100     00524                                  /DMTAPEC1
0002 FILE2                  0001     77315        77315        TAPE3        V         B         00020     00100                                  /DMTAPEC1
0003 FILE3                  0001     77315        77315        TAPE3        F         B         00030     00090                                  /DMTAPEC1
0004 FILE4                  0001     77315        77315        TAPE3        F         B         00040     00080                                  /DMTAPEC1
0005 FILE5                  0001     77315        77315        TAPE3        F         B         00050     00050                                  /DMTAPEC1

*****     TAPE LABEL DISPLAY          DATE-11/11/77          TIME-00.02.45     *****
ASCII TAPE LABEL          UNIT-T4          FILE SERIAL-TAPE4          OWNER CODE

----------HEADER 1----------                    ----------HEADER 2----------
FILE SEQ  FILE-IDENTIFIER   VOL SEQ  CREATE DATE  EXPIRE DATE  FILE SERIAL  REC FORM  BLK ATTR  REC LENG  BLOCK LENG  RECORDING TECHNIQUE  PRTR CNTRL  JOBNAME  STEPNAME
0001 FILE1                  0001     77315        77315        TAPE4        U         B         00100     00524                                  /DMTAPEC1
0002 FILE2                  0001     77315        77315        TAPE4        U         B         00020     00100                                  /DMTAPEC1
0003 FILE3                  0001     77315        77315        TAPE4        F         B         00030     00090                                  /DMTAPEC1
0004 FILE4                  0001     77315        77315        TAPE4        F         B         00040     00080                                  /DMTAPEC1
0005 FILE5                  0001     77315        77315        TAPE4        F         B         00050     00050                                  /DMTAPEC1
```

MESSAGES FOR TAPE INITIALIZATION

Note: The following message is printed if the printer is not allocated to another partition.

Message	Meaning
INITIALIZATION ON xx COMPLETE	This message is printed when initialization of a tape is complete. xx indicates the unit (T1, T2, T3, or T4) on which the initialization is complete.

MEANING OF VOLUME LABEL INFORMATION

Heading	Meaning
Volume Label	
FILE SERIAL	The volume serial number (from the REEL parameter).
OWNER CODE	Additional identification (from the ID parameter).
Header 1 Label	
FILE SEQ	Sequence number of the file on the volume.
FILE-IDENTIFIER	The filename of the file on tape. This is the name from the LABEL parameter of the OCL FILE statement when the file was created.
VOL SEQ	The sequence number of this volume in a multivolume file.
CREATE DATE	The date this file was created. This is a Julian date. The format is yyddd where yy is the last two digits of the year and ddd is the day in the year. Example: 77315 = the 315th day of 1977 or November 11, 1977.
EXPIRE DATE	The date this file expires. This Julian date is the creation date plus the number of days specified by the RETAIN parameter on the OCL FILE statement.
FILE SERIAL	The serial number of the tape volume. This is the same as the SERIAL field in the volume label. However, for multivolume files, it is the same as the SERIAL field of the first volume's volume label.
Header 2 Label	
REC FORM	The record format of this file (from the RECFM parameter on the OCL FILE statement when this file was created). The formats are: F – Fixed length V – Variable length U – Undefined length
BLK ATTR	Block attributes: B – Blocked records S – Spanned records R – Blocked and spanned records blank – Neither blocked nor spanned *Note:* Spanned records cannot be created on System/3.

Heading	Meaning
REC LENG	Record length (from the RECL parameter on the OCL FILE statement when this file was created).
BLOCK LENG	Block length (from the BLKL parameter on the OCL FILE statement when this file was created).
RECORDING TECHNIQUE	T – Odd parity with translation C – Odd parity with conversion E – Even parity without translation ET – Even parity with translation blank – Odd parity without translation or conversion
PRTR CNTRL	Printer control character. This field is blank on tapes created on System/3. For tapes created on other systems, the characters are: A – ASCII control characters M – Machine control characters blank – No control characters
JOBNAME/STEPNAME	Identifies the job that created the tape. The JOBNAME field contains the job name and the STEPNAME field contains the name of the step within the job. The JOBNAME is blank if JOBNAME is the same as STEPNAME.

Tape Error Summary Program—$TVES

PROGRAM DESCRIPTION

The tape error summary program has the following function:

Display magnetic tape errors

The IBM System/3 Model 15 keeps an account of errors that occur on the tape drives. This error information is stored on one of the customer engineer tracks located on the volume from which an IPL is performed. The Tape Error Summary program provides a summary, by volume and by unit, of temporary read and write errors.

This program does not reset the disk area that contains the error statistics. Therefore, you should periodically run $TVES to prevent the overlaying of data on the track. (The system will not issue a message if the data is overlaid.)

There are no control statements necessary for this program. After being loaded from the program or system pack, the tape error summary program reads the data from the disk and sorts it by volume and unit. When all the data is read or the available main storage is filled, the error data is printed. If no tape errors are recorded, the message THERE ARE NO VALID TAPE ERRORS LOGGED is printed.

ERROR LOGGING FORMAT

```
SUMMARY MAGNETIC TAPE ERROR STATISTICS BY VOLUME  DATE 03/27/76
   (1)          (2)        (3)        (4)        (5)
VOLUME      SIO        TEMP       TEMP       WRITE
SERIAL      COUNT      READ       WRITE      SKIP

T1          06512      0000       0028       0028
TAPE1       00016      0000       0001       0001
TAPE3       00021      0000       0001       0001


SUMMARY MAGNETIC TAPE ERROR STATISTICS BY TAPE UNIT  DATE 03/27/76
               (2)        (3)        (4)        (5)        (6)
TAPE        SIO        TEMP       TEMP       WRITE      DIAG
UNIT        COUNT      READ       WRITE      SKIP       TRACK

T1          06528      0000       0029       0029       0000
T4          00021      0000       0001       0001       0000
```

(1) For any file that has more than two volumes on a unit ,,,,,, is printed as the volume serial for all volumes on that unit except the last volume.

For a tape that is not being used by tape data management ,,,,,, is printed as the volume serial.

For non-labeled tapes ****** is printed as the volume serial. For tapes with nonstandard labels, NS is printed as the volume serial.

(2) The number of tape operations performed (SIO means Start I/O).

(3) Temporary read errors.

(4) Temporary write errors.

(5) Write skips caused by temporary write errors.

(6) Diagnostic track errors. This is used by IBM customer engineers.

OCL CONSIDERATIONS

The following OCL statements are needed to load the tape error summary program.

```
// LOAD $TVES,code
// RUN
```

The code you supply depends on the location of the simulation area containing the program. Possible codes are R1, F1, R2, F2.

VTOC Service Program—$WVTOC

PROGRAM DESCRIPTION

The VTOC service program has the following function:

Compress a VTOC

For possible improved performance in file access time, all file directory entries are moved to the beginning of the VTOC (volume table of contents).

CONTROL STATEMENT SUMMARY

The control statements you must supply depend on the desired results.

Function	Control Statements
Move all active file directory entries to the beginning of the VTOC	// COMPRESS PACK-name,UNIT-code // END

PARAMETER SUMMARY

Parameter	Description
PACK-name	Name of the main data area
UNIT-code	Location of the area. Possible codes are those for the main data areas.

PARAMETER DESCRIPTIONS

PACK Parameter

The PACK parameter (PACK-name) tells the program the name of the main data area on which the VTOC is to be compressed. The VTOC service program compares the name in the PACK parameter with the volume label of the area to ensure that they match.

UNIT Parameter

The UNIT parameter (UNIT-code) tells the program the location of the VTOC to be compressed. Possible codes are those for the main data areas.

OCL CONSIDERATIONS

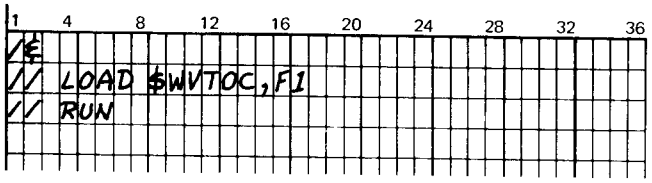
The following OCL statements are needed to load the VTOC service program:

```
// LOAD $WVTOC,code
// RUN
```

The code you supply depends on the location of the simulation area containing the VTOC service program. Possible codes are R1, F1, R2, F2.

EXAMPLES

Figure 4-118 shows the OCL statements needed to load the VTOC service program.

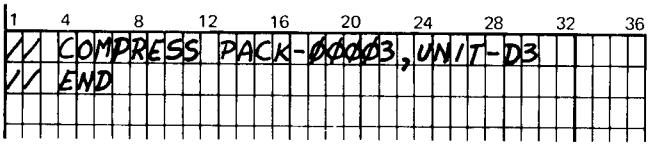


Explanation:

VTOC service program is loaded from F1.

Figure 4-94. OCL Load Sequence for VTOC Service Program

Figure 4-95 shows the control statements needed to compress a VTOC.



Explanation:

- The main data area containing the VTOC to be compressed is named 00003 (PACK-00003 in the COMPRESS statement).
- The main data area containing the VTOC to be compressed is on volume 1 of drive 3 (UNIT-D3).

Figure 4-119. Control Statements for Compressing a VTOC

Character	Hexadeciaml Equivalent
Blank	40
¢	4A
.	4B
<	4C
(	4D
+	4E
|	4F
&	50
!	5A
$	5B
*	5C
)	5D
;	5E
¬	5F
- (minus)	60
/	61
,	6B
%	6C
_ (underscore)	6D
>	6E
?	6F
:	7A

Character	Hexadecimal Equivalent
#	7B
@	7C
' (apostrophe)	7D
=	7E
"	7F
A	C1
B	C2
C	C3
D	C4
E	C5
F	C6
G	C7
H	C8
I	C9
}	D0 (1)
J	D1
K	D2
L	D3
M	D4
N	D5
O	D6
P	D7

Character	Hexadecimal Equivalent
Q	D8
R	D9
S	E2
T	E3
U	E4
V	E5
W	E6
X	E7
Y	E8
Z	E9
0	F0
1	F1
2	F2
3	F3
4	F4
5	F5
6	F6
7	F7
8	F8
9	F9

(1) On the CRT a X'D0' displays as an &.

The factors that control file size are explained in the *IBM System/3 Disk Concepts and Planning Guide,* GC21-7571. The structure of a file (sequential, direct, or indexed) on the main data area of a 3340 or 3344 is virtually identical to the structure of a file on the 5445. The main data area on a 3340 consists of 48 records per track and 166 cylinders; each main data area on a 3344 consists of 48 records per track and 186 cylinders; the 5445 consists of 20 records (sectors) per track and 200 cylinders per disk pack.

Following is a list of similarities between the 5445, 3340, and 3344:

- Index entries are calculated using the following formula: key length + 4.
- Index and data areas are contained on whole tracks.
- Indexed files with indexes of 16 tracks or more have space reserved for a track index.
- Spanned records are used.
- The index area must have at least one extra record (sector) for a delimiter or at least three extra records (sectors) if additions will be made to the file.
- There are 20 tracks per cylinder (3980 tracks on the 5445; 3320 tracks of main data area on the 3340; 3720 tracks of main data area on the 3344).

The following tables and discussion include the specific information necessary to evaluate file size on the 3340 or 3344 main data area.

DATA AREA TRACK REQUIREMENTS

Sequential, direct, and indexed data areas are all structured the same. Figure B-1 shows the number of tracks needed to store a given number of logical records, using various logical record lengths.

NUMBER OF TRACKS[1]

Number of Logical Records	Logical Record Length 50		Logical Record Length 64		Logical Record Length 100		Logical Record Length 128		Logical Record Length 256	
	5445	3340 and 3344	5445	3340 and 3344	5445	3340 and 3344	5445	3340 and 3344	5445	3340 and 3344
500	5	3	7	3	10	5	13	6	25	11
1000	10	5	13	6	20	9	25	11	50	21
1500	15	7	19	8	30	13	38	16	75	32
2000	20	9	25	11	40	17	50	21	100	42
2500	25	11	32	14	49	21	63	27	125	53
3000	30	13	38	16	59	25	75	32	150	63
3500	35	15	44	19	69	29	88	37	175	73
4000	40	17	50	21	79	33	100	42	200	84
4500	44	19	57	24	88	37	113	47	225	94
5000	49	21	63	27	98	41	125	53	250	105
5500	54	23	69	29	108	45	138	58	275	115
6000	59	25	75	32	118	49	150	63	300	125
6500	64	27	82	34	127	53	163	68	325	136
7000	69	29	88	37	137	57	175	73	350	146
7500	74	31	94	40	147	62	188	79	375	157
8000	79	33	100	42	157	66	200	84	400	167
8500	84	35	107	45	167	70	213	89	425	178
9000	88	37	113	47	176	74	225	94	450	188
9500	93	39	119	50	186	78	238	99	475	198
10,000	98	41	125	53	196	82	250	105	500	209
10,500	103	43	132	55	206	86	263	110	525	219
11,000	108	45	138	58	215	90	275	115	550	230
11,500	113	47	144	60	225	94	288	120	575	240
12,000	118	49	150	63	235	98	300	125	600	250
12,500	123	51	157	66	245	102	313	131	625	261
13,000	127	53	163	68	254	106	325	136	650	271
13,500	132	55	169	71	264	110	338	141	675	282
14,000	137	57	175	73	274	114	350	146	700	292
14,500	142	60	182	76	284	119	363	152	725	303
15,000	147	62	188	79	293	123	375	157	750	313
15,500	152	64	194	81	303	127	388	162	775	323
16,000	157	66	200	84	313	131	400	167	800	334
16,500	162	68	207	86	323	135	413	172	825	344
17,000	167	70	213	89	333	139	425	178	850	355
17,500	171	72	219	92	342	143	438	183	875	365
18,000	176	74	225	94	352	147	450	188	900	375
18,500	181	76	232	97	362	151	463	193	925	386
19,000	186	78	238	99	372	155	475	198	950	396
19,500	191	80	244	102	381	159	488	204	975	407
20,000	196	82	250	105	391	163	500	209	1000	417
21,000	206	86	263	110	411	171	525	219	1050	438
22,000	215	90	275	115	430	180	550	230	1100	459

[1]The number of tracks for data records does not include indexes. For the 3340 and 3344, this is the main data area, not the simulation area.

Figure B-1 (Part 1 of 2). Disk Requirements for Data Records

NUMBER OF TRACKS[1]

Number of Logical Records	Logical Record Length 50		Logical Record Length 64		Logical Record Length 100		Logical Record Length 128		Logical Record Length 256	
	5445	3340 and 3344	5445	3340 and 3344	5445	3340 and 3344	5445	3340 and 3344	5445	3340 and 3344
23,000	225	94	288	120	450	188	575	240	1150	480
24,000	235	98	300	125	469	196	600	250	1200	500
25,000	245	102	313	131	489	204	625	261	1250	521
26,000	254	106	325	136	508	212	650	271	1300	542
27,000	264	110	338	141	528	220	675	282	1350	563
28,000	274	114	350	146	547	228	700	292	1400	584
29,000	284	119	363	152	567	237	725	303	1450	605
30,000	293	123	375	157	586	245	750	313	1500	625
31,000	303	127	388	162	606	253	775	323	1550	646
32,000	313	131	400	167	625	261	800	334	1600	667
33,000	323	135	413	172	645	269	825	344	1650	688
34,000	333	139	425	178	665	277	850	355	1700	709
35,000	342	143	438	183	684	285	875	365	1750	730
36,000	352	147	450	188	704	293	900	375	1800	750
37,000	362	151	463	193	723	302	925	386	1850	771
38,000	372	155	475	198	743	310	950	396	1900	792
39,000	381	159	488	204	762	318	975	407	1950	813
40,000	391	163	500	209	782	326	1000	417	2000	834
41,000	401	167	513	214	801	334	1025	428	2050	855
42,000	411	171	525	219	821	342	1050	438	2100	875
43,000	420	175	538	224	840	350	1075	448	2150	896
44,000	430	180	550	230	860	359	1100	459	2200	917
45,000	440	184	563	235	879	367	1125	469	2250	938
46,000	450	188	575	240	899	375	1150	480	2300	959
47,000	459	192	588	245	918	383	1175	490	2350	980
48,000	469	196	600	250	938	391	1200	500	2400	1000
49,000	479	200	613	256	958	399	1225	511	2450	1021
50,000	489	204	625	261	977	407	1250	521	2500	1042
75,000	733	306	938	391	1465	611	1875	782	3750	1563
100,000	977	407	1250	521	1954	814	2500	1042	5000	2084
125,000	1221	509	1563	652	2442	1018	3125	1303	6250	2605
150,000	1465	611	1875	782	2930	1221	3750	1563	7500	3125
175,000	1709	713	2188	912	3418	1425	4375	1823	8750	3646
200,000	1954	814	2500	1042	3907	1628	5000	2084	10,000	4167

[1] The number of tracks for data records does not include indexes. For the 3340 and 3344 this is the main data area, not the simulation area.

Figure B-1 (Part 2 of 2). Disk Requirements for Data Records

INDEX AREA TRACK REQUIREMENTS

Indexed files also require tracks for the file index. The following chart shows how many keys can be contained on one track of the file index.

Key-Length	Number of Keys Per Index Track 5445	3340 and 3344
1	1020	2448
2	840	2016
3	720	1728
4	640	1536
5	560	1344
6	500	1200
7	460	1104
8	420	1008
9	380	912
10	360	864
11	340	816
12	320	768
13	300	720
14	280	672
15	260	624
16	240	576
17	240	576
18	220	528
19	220	528
20	200	480
21	200	480
22	180	432
23	180	432
24	180	432
25	160	384
26	160	384
27	160	384
28	160	384
29	140	336

Since there is one key for each logical data record in the file, the number of tracks required for the file index is easily calculated for a given number of logical records.

The index area also includes a disk track index if the file index requires 16 or more tracks. Figure B-2 shows the file sizes supported by one track of the disk track index for various key lengths.

Key-Length	File Capacity (Number of Logical Records) 5445	3340 and 3344
1	1,040,400	5,992,704
2	705,600	4,064,256
3	518,400	2,985,984
4	409,600	2,359,296
5	313,600	1,806,336
6	250,000	1,440,000
7	211,600	1,218,816
8	176,400	1,016,064
9	144,400	831,744
10	129,600	746,496
11	115,600	665,856
12	102,400	589,824
13	90,000	518,400
14	78,400	451,584
15	67,600	389,376
16	57,600	331,776
17	57,600	331,776
18	48,400	278,784
19	48,400	278,784
20	40,000	230,400
21	40,000	230,400
22	32,400	186,624
23	32,400	186,624
24	32,400	186,624
25	25,600	147,456
26	25,600	147,456
27	25,600	147,456
28	25,600	147,456
29	19,600	112,896

Figure B-2. File Capacity for One Track of Disk Track Index

TRACK USAGE FOR INDEX FILES

When loading an indexed file, you specify either the number of records in the file or the number of tracks. When you specify the number of tracks, the system determines how the specified space is to be split among data tracks, file index tracks, and disk track index tracks. Figure B-3 illustrates how the system splits an area on the 3340 or 3344 when the TRACKS parameter is used in the OCL statement, and illustrates the maximum record capacities for various size indexed files. Remember to use the smaller of the two numbers: *number of keys, or number of logical data records.*

Number of Tracks	Key Length	Logical Record Length	Disk Track Index	Number of File Index Tracks	Number of Data Tracks	Number of Keys	Number of Logical Data Records
5	5	64		1	4	1,316	768
5	5	128		1	4	1,316	384
5	5	256		1	4	1,316	192
5	10	64		1	4	846	768
5	10	128		1	4	846	384
5	10	256		1	4	846	192
10	5	64		2	8	2,660	1,536
10	5	128		1	9	1,316	864
10	5	256		1	9	1,316	432
10	10	64		2	8	1,710	1,536
10	10	128		1	9	846	864
10	10	256		1	9	846	432
50	5	64		7	43	9,380	8,256
50	5	128		4	46	5,348	4,416
50	5	256		2	48	2,660	2,304
50	10	64		9	41	7,758	7,872
50	10	128		5	45	4,302	4,320
50	10	256		3	47	2,574	2,256
100	5	64		13	87	17,444	16,704
100	5	128		7	93	9,380	8,928
100	5	256		4	96	5,348	4,608
100	10	64	1	18	81	15,534	15,552
100	10	128		10	90	8,622	8,640
100	10	256		6	94	5,166	4,512
500	5	64	1	63	436	84,644	83,712
500	5	128	1	34	465	45,668	44,640
500	5	256	1	18	481	24,164	23,088
500	10	64	1	91	408	78,606	78,336
500	10	128	1	50	449	43,182	43,104
500	10	256	1	27	472	23,310	22,656
1,000	5	64	1	125	874	167,972	167,808
1,000	5	128	1	67	932	90,020	89,472
1,000	5	256	1	35	964	47,012	46,272
1,000	10	64	1	182	817	157,230	156,864
1,000	10	128	1	100	899	86,382	86,304
1,000	10	256	1	53	946	45,774	45,408
2,000	5	64	1	250	1,749	335,972	335,808
2,000	5	128	1	134	1,865	180,068	179,040
2,000	5	256	1	69	1,930	92,708	92,640
2,000	10	64	1	364	1,635	314,478	313,920
2,000	10	128	1	200	1,799	172,786	172,704
2,000	10	256	1	106	1,893	91,566	90,864
3,000	5	64	1	375	2,624	503,972	503,808
3,000	5	128	1	200	2,799	268,772	268,704

Figure B-3 (Part 1 of 2). Sample Record Capacities of Indexed Files

Number of Tracks	Key Length	Logical Record Length	Disk Track Index	Number of File Index Tracks	Number of Data Tracks	Number of Keys	Number of Logical Data Records
3,000	5	256	1	104	2,895	139,748	138,960
3,000	10	64	1	546	2,453	471,726	470,976
3,000	10	128	1	300	2,699	259,182	259,104
3,000	10	256	1	158	2,841	136,494	136,368
3,320	5	64	1	415	2,904	557,732	557,568
3,320	5	128	1	222	3,097	298,340	297,312
3,320	5	256	1	115	3,204	154,532	153,792
3,320	10	64	1	604	2,715	521,838	521,280
3,320	10	128	1	332	2,987	286,830	286,752
3,320	10	256	1	175	3,144	151,182	150,912

Figure B-3 (Part 2 of 2). Sample Record Capacities of Indexed Files

CORE INDEX

The core index is a table of pointers to the index tracks of an indexed file. The core index is built just before your COBOL or RPG II program is executed and contains, at most, one entry for each index track. Figure B-4 shows the number of bytes of main storage required for a core index that provides the most efficient random processing of an indexed file, using key length and number of logical records as variables.

Key Length	(Number of Logical Records (in 1000s) 2	5	8	10	15	20	50
4	12	24	36	42	60	84	198
5	14	28	42	56	84	105	266
6	16	40	56	72	104	136	336
7	18	45	72	90	126	171	414
8	20	50	80	100	150	200	500
9	33	66	99	121	187	242	605
10	36	72	120	144	216	288	696
11	39	91	130	169	247	325	806
12	42	98	154	196	280	378	924
13	45	105	180	210	315	420	1,050
14	48	128	192	240	368	480	1,200
15	68	153	221	289	425	561	1,377
16	72	162	252	324	486	630	1,566
17	76	171	266	342	513	665	1,653
18	80	200	320	280	580	760	1,900
19	84	210	336	399	609	798	1,995
20	110	242	374	462	704	924	2,310

Figure B-4. Core Index Sizes for 3340 and 3344 Single Volume Indexed Files

CALCULATING FILE SIZES (MAIN DATA AREA)—SUMMARY

This section contains step-by-step explanations of some common calculations of file sizes

Determining the Number of Tracks in a Sequential or Direct File

1. Number of logical records x logical record length
   = number of characters

2. Number of characters (from step 1) / 12,288 (number of characters/track)
   = number of tracks (round to the next highest whole number)

Determining the Number of Tracks in an Indexed File (Main Data Area)

To determine the number of data tracks in an indexed file, the following two steps should be used:

1. Number of logical records x logical record length
   = number of characters

2. Number of characters (from step 1) / 12,288 (number of characters/track)
   = number of data tracks (round to the next highest whole number)

The following four steps should then be followed to determine the number of file index tracks in an indexed file:

1. Key length + 4 = index length

2. 256 (number of characters/record) / Index length (from step 1)
   = number of entries per record (drop remainder)

3. Number of logical records / Number of entries per record (from step 2)
   = number of records (round to the next highest whole number; then, add one record for a delimiter, and two or more additional records if you plan to add logical records to the file later)

4. Number of records (from step 3) / 48 (number of records/track)
   = number of index tracks (round to the next highest whole number)

Determining the Number of Tracks of Disk Track Index

If an indexed file has more than 15 index tracks (from step 4 in the preceding calculation), the file will have a disk track index in addition to the file index. The following two steps should be used to determine the number of tracks needed for the disk track index:

1. Number of entries per record (from step 2 of preceding calculation)
   x number of records per track (48)
   = keys per track

2. Number of index tracks (if greater than 15) / Keys per track
   = number of disk track index tracks (round to the next highest whole number)

The total number of tracks in an indexed file can be determined by adding the number of data tracks, the number of file index tracks, and the number of disk track index tracks.

Converting Cylinder/Track to Track Number

To convert cylinder/track to track number, multiply cylinder number by the number of tracks on each cylinder and add track number.

Examples:

Simulation area

6/1 = cylinder/track
6 x 2 + 1 = 13
13 = track number

Main data area on 3340 and 3344

6/1 = cylinder/track
6 x 20 + 1 = 121
121 = track number

Converting Track Number of Cylinder/Track

To convert track number to cylinder/track, divide track number by the number of tracks on a cylinder. The quotient is the cylinder and the remainder is the track.

Examples:

Simulation area

13 = track number
13 ÷ 2 = 6 (remainder 1)
6/1 is the cylinder/track

Main data area on 3340 and 3344

121 = track number
121 ÷ 20 = 6 (remainder 1)
6/1 is the cylinder/track

Appendix C. Operator Control Commands (OCC)

OCC SUMMARY

All operator control commands (OCC) are entered from the CRT/keyboard. By using OCC, you can communicate with the system before, during, or after job execution. You can also use the OCC function to enter comments in the system history area. Each comment is identified by an asterisk in position 1. A maximum of 40 characters, including the asterisk, can be entered. For a complete description of OCC, see the *IBM System/3 Model 15 Operator's Guide,* GC21-5075.

INFORMATION ABOUT SYNTAX ILLUSTRATIONS

- Uppercase letters, numbers, and commas must be entered as shown. Parentheses indicate an abbreviation for the item above. Do not enter brackets, braces, or parentheses.
- Lowercase items represent variables.
- Items or groups of items within brackets [ ] are optional—enter one or none.
- Items in braces { } are alternatives—enter one.
- Optional items that are underlined indicate the default value. If none of the items is entered, the default value is used.
- All operands are positional. Operands must be entered in the order shown. If any operand (except the last) is omitted, a comma must be entered in its place.
- There must be at least one blank between the command and the operands. The first blank after the operands start terminates the operands. There must be only blanks after the last operand.

Command	Operands	Function
CANCEL (CN)	P1 [2/3] P2 P3	Cancels program execution in a partition. The 2 or 3 following the P1, P2, or P3 indicates the option of the cancel.
	SPOOL (SP)	Cancels spooling support (at IPL only).
	*RDRQ (*RQ) *PRTQ (*WQ) *PCHQ (*PQ)	Cancels the entire queue.
	RDRQ, (RQ) {jobname / characters**}	Cancels a job or job step, or multiple jobs or job steps.
	PRTQ, (WQ) {jobname / characters**} [,stepname / ,characters**]	
	PCHQ, (PQ) {jobname / characters**} [,stepname / ,characters**]	

Command	Operands	Function
CHANGE (G)	CORE,size, [RDRQ / (RQ)] , {jobname / characters**} CPY,[number], (PRTQ / (WQ) / PCHQ / (PQ)) , {jobname / characters**} [,stepname / ,characters**] CRD,card type, [PCHQ / (PQ)] , {jobname / characters**} [,stepname / ,characters**] FRM,forms type, [PRTQ / (WQ)] , {jobname / characters**} [,stepname / ,characters**] PTN, [1 / 2 / 3 / A / B / C / D] , [RDRQ / (RQ)] ,jobname PTY, (1 / 2 / 3 / 4 / 5) , RDRQ, {jobname / characters**} PRTQ, {jobname / characters**} [,stepname / ,characters**] PCHQ, {jobname / characters**} [,stepname / ,characters**]	Changes the main storage requirements, number of copies, card type, forms type, partition assignment, or job priority of a job or job step or multiple jobs or job steps on a queue.
DATE (DT)	{mm/dd/yy / mmddyy / dd/mm/yy / ddmmyy}	Sets the system and partition dates.

Command	Operands	Function
DISPLAY (D)	STATUS [,P1] [,P2] [,P3] [,N] HISTORY[1] (H) [,task] [,task] ... SVAID (SV) CORE RDRQ (RQ) PRTQ (WQ) PCHQ (PQ)	Displays the system status, partition status, and area names. Displays the history of the entire system or of one through seven tasks. Displays service aids options. Displays main storage. Displays the contents of the spooling queues.
DUMP (K)	P1 [2/3] P2 P3 SYSTEM [,DISK] (S)	Dumps any partition or all of main storage to a printer. Or, dumps all of main storage, the SWA, part of the SHA, the configuration records, the PTF log sectors, microcode levels, the error recording data, and the saved transient area to the $SYSDUMP file. (The transient area is included only if a process check occurred in the transient area and the DUMP command is given before end of job for that partition.)

[1] Possible task operands are P1, P2, P3, S, SP, OCC, CCP, CCS, CCU, C44, C55, C66, C77, C88, C99, CCC, CDD, CEE, CGG, CHH, CMM, CPP, CTT, CUU, CVV, CWW, CXX, CYY, CZZ. The meaning of each task operand is included in the *IBM System/3 Model 15 Operator's Guide*, GC21-5075.

Command	Operands	Function
HALT (HT)	[P1 P2 P3 SHA [,CCP] [,tracks]]	Causes the partition to halt with an EJ or ES message after the current job step is complete. Causes the system to halt or $HACCP to be automatically invoked before unprinted system history area entries are overlaid.
HIPTY (HP)	high-priority	Limits the priority of jobs placed on the reader queue by $QCOPY under CCP.
HOLD (H)	RDRQ [,jobname ,characters**] (RQ) PRTQ [,jobname [,stepname] ,characters** ,characters**] (WQ) PCHQ [,jobname [,stepname] ,characters** ,characters**] (PQ)	Prevents a job, a job step, or an entire queue from being scheduled for input or output.
IDELETE (I)		Causes the system to automatically delete I-type (informational) messages to which there was no response (when there should have been a response).
KEEP (KP)	RDRQ, {jobname (RQ) {characters**} PRTQ, {jobname} [,stepname] (WQ) {characters**} [,characters**] PCHQ, {jobname} [,stepname] (PQ) {characters**} [,characters**]	Keeps single or multiple printed, punched, and executed job steps and/or jobs in the hold state on the queue.
NOHALT (NHT)	[P1 P2 P3 SHA]	Changes the processing mode of a partition from halt to no-halt or allows system history area entries to be overlaid without operator notification.

Command	Operands	Function
NOIDELETE (NI)		Causes the system to cease automatic deletion of I-type messages that received no response.
PTY (Y)	{ SPOOL (SP) / P1 / P2 / P3 } { = / - } ...	Changes the priority for the system tasks: spooling (reader, print writer, punch writer), partition 1, partition 2, and partition 3.
READER (RDR)	[P1 / P2 / P3] ,device	Changes the system input device for a partition.
RELEASE (R)	{ RDRQ (RQ) [,jobname / ,characters**] / PRTQ (WQ) [,jobname / ,characters** [,stepname / ,characters**]] / PCHQ [,jobname / ,characters** [,stepname / ,characters**]] }	Releases a job or job step, multiple jobs or job steps, or an entire queue from the hold state.
RESTART (T)	[PRT (W) / PR1 (W1) / PR2 (W2) / PCH (P) [,PAGE (PG) ,number]]	Restarts a job step's printed or punched output. PR1 (W1) and PR2 (W2) are valid operands only if a second 1403 Printer is installed.
REUSE	{ PRTQ / PCHQ ,jobname[,stepname] }	Causes track groups that have been printed or punched to be made available. (Allows the area to be reused.)

Command	Operands	Function
SET	[P1 / P2 / P3 / SH] ,size	Overrides the partition and file share area size specified during system generation.
START (S)	SPOOL (SP) [,P1 / ,P2 / ,P3 / [,NEW] [,unit] ,size][,track group size]]	Restarts spooling in the designated partition.
		Starts spooling at IPL.
	RDR[,ROLL] (R)	Starts the spooled reader, print writer, or punch writer.
	PRT (W) / PR1 (W1) / PR2 (W2) / PCH (P) [,forms type]	PR1 (W1) and PR2 (W2) are valid operands only if a second 1403 Printer is installed.
	P1 / P2 / P3	Restarts program execution in a partition.
STOP (P)	SPOOL (SP) {,P1 / ,P2 / ,P3}	Terminates spooling in specified partition.
	RDR (R)	Stops the spooled reader, print writer, or punch writer.
	PRT (W) / PR1 (W1) / PR2 (W2) [,STEP (ES) / ,PAGE (PG)]	PR1 (W1) and PR2 (W2) are valid operands only if a second 1403 Printer is installed.
	PCH (P) [,STEP (ES)]	
	P1 / P2 / P3	Stops program execution in a partition.

Command	Operands	Function
TIME (TM)	{ hh/mm/ss { hhmmss	Sets time-of-day clock.
TLOG (TL)	{ ON { OFF	Enables or disables the transaction logging routine (this routine must be currently loaded).
TRACE (E)	{ ON { OFF { ,SYSTEM { (S)	Enables or disables the system trace routine (which must be currently loaded).

Appendix D. SUBR15—Library Entry Retrieval Subroutine

The library entry retrieval subroutine can be incorporated into a user-written program to retrieve a single entry or a group of entries from any library. The user supplies such information as the name of the entry, the unit containing the library, and the library to be accessed.

Library entries are passed to the user program one record at a time. The first record returned to the user program by SUBR15 for each entry is an appropriate COPY statement; the last record for each entry is a CEND statement. The format of the records produced by SUBR15 is the same as that produced by the copy library-to-card function of $MAINT. If the records are written to a disk file, that file can be processed by the copy file-to-library function of $MAINT.

Note: If SUBR15 is used in a program executing under control of the communications control program (CCP), the user must ensure that the library to be used is not being changed by another program while SUBR15 is reading from it.

LINKING SUBR15 WITH RPG II

SUBR15 is linked to a user RPG II program via an EXIT statement followed by seven RLABL statements. (For subroutine linkage information, see the *IBM System/3 RPG II Reference Manual,* SC21-7504.) These RLABL statements must be specified in the following order:

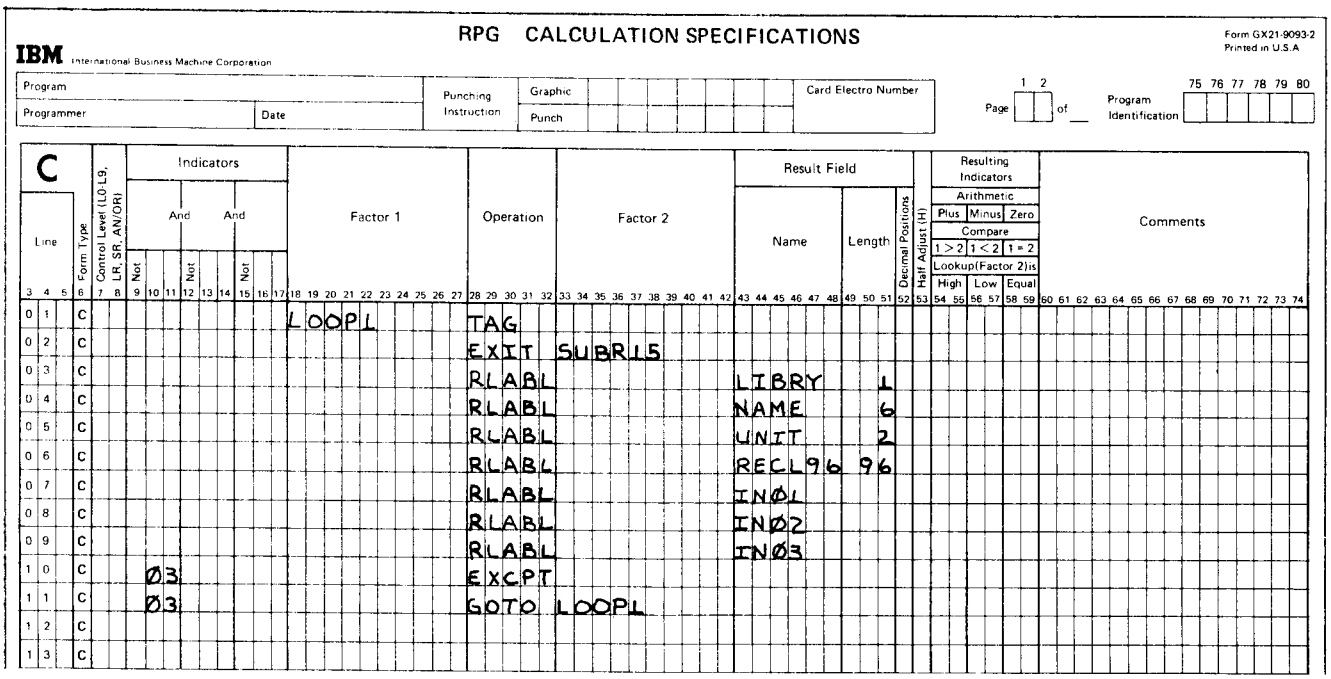
Sequence	Operation Entry	Result Field Entry	Description
1	RLABL	Field	The name of a 1-character alphameric RPG II field that contains a code identifying the library to be used. The code in the field can be: S – Source P – Procedure O – Object R – Routine
2	RLABL	Field	The name of a 6-character alphameric RPG II field that contains the name of the library entry to be retrieved. If this field contains five or fewer characters followed by a period ('ccccc.'), all entries having names beginning with those characters are copied. (This is similar to the 'characters.ALL' capability of $MAINT.) If the first character of this field is a period, then all entries in that library are copied. (This is similar to the NAME-ALL capability of $MAINT.)
3	RLABL	Field	The name of a 2-character alphameric RPG II field that contains a code identifying the unit location of the library. The code in the field can be: R1, F1, R2, F2.
4	RLABL	Field	The name of an 80- or 96-character alphameric RPG II field that will contain each record processed by SUBR15.

Sequence	Operation Entry	Result Field Entry	Description
5	RLABL	INnn	A unique RPG II indicator to be set on if the library entry is not found or if the library does not exist on the specified unit.
6	RLABL	INnn	A unique RPG II indicator to be set on when the request for retrieval is complete. (When this indicator is on, the RPG II program can request another entry or entries.)
7	RLABL	INnn	A unique RPG II indicator to be set on when a record is passed to the RPG II program by SUBR15.

Note: The three indicators are set off by SUBR15 upon receiving control from the user program.

Example

The following example shows how an RPG II program uses SUBR15:



RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Form GX21-9093-2 Printed in U.S.A.

Program | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number | Page 1 2 of __ | Program Identification 75 76 77 78 79 80

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / Not	And / Not	And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup(Factor 2)is: High / Low / Equal	Comments
01	C					LOOP1	TAG							
02	C						EXIT	SUBR15						
03	C						RLABL		LIBRY	1				
04	C						RLABL		NAME	6				
05	C						RLABL		UNIT	2				
06	C						RLABL		RECL96	96				
07	C						RLABL		IN01					
08	C						RLABL		IN02					
09	C						RLABL		IN03					
10	C		03				EXCPT							
11	C		03				GOTO	LOOP1						
12	C													
13	C													

- The LIBRY field contains a code (S, P, O, or R) that identifies the library.
- The NAME field contains the name of the library entry or entries to be retrieved.
- The UNIT field contains a code (R1, F1, R2, or F2) that identifies the location of the library.
- The RECL96 field is a 96-byte field that contains each record processed by SUBR15.
- Indicator 01 (IN01) is set on if the library entry is not found or if the library does not exist on the specified unit.
- Indicator 02 (IN02) is set on when the retrieval request is complete.
- Indicator 03 (IN03) is set on when a record is passed to the RPG II program by SUBR15.
- Indicator 03 is used to condition (via the EXCPT operation code) writing the record to a file, and to condition the branch to LOOP1 to request another record.

LINKING SUBR15 WITH ASSEMBLER

Linkage to SUBR15 from an assembler language program is via a branch instruction (to SUBR15) followed by a series of parameters. An EXTRN statement with an operand of SUBR15 is also needed. When the subroutine is entered, the value in index register 1 (XR1) must be the address from which the displacement(s) for parameters 5, 6, and 7 are determined. (Linkage of SUBR15 to the assembler program is accomplished by the Overlay Linkage Editor of the SCP. For further information about assembler linkage, refer to the *IBM System/3 Basic Assembler Reference Manual,* SC21-7509, and to the *IBM System/3 Overlay Linkage Editor Reference Manual,* GC21-7561.)

The series of parameters can be coded as follows:

Sequence	Operation Entry	Operand Entry	Description
1	DC DC	IL1'0' AL2(address)	The address of a 1-byte field that contains a code identifying the library to be used: S – Source P – Procedure O – Object R – Routine
2	DC DC	IL1'5' AL2(address)	The address of the rightmost byte of a 6-byte field that contains the name of the library entry to be retrieved. If this field contains five or fewer characters followed by a period ('ccccc.'), all entries having names beginning with the indicated characters are copied. (This is similar to the 'characters.ALL' capability of $MAINT.) If the first character of this field is a period, then all entries in that library are copied. (This is similar to the NAME-ALL capability of $MAINT.)
3	DC DC	IL1'1' AL2(address)	The address of the rightmost byte of a 2-byte field that contains the unit code of the library. The code in the field can be: R1, F1, R2, or F2.
4	DC DC or DC DC	IL1'79' AL2(address) [for 80-char field] IL1'95' AL2(address) [for 96-char field]	The address of the rightmost byte of an 80- or 96-character field that will contain each record processed by SUBR15.

Sequence	Operation Entry	Operand Entry	Description
5	DC DC DC	XL1'00' XL1'mask' XL1 'displacement'	Indicates one of the following errors: desired library entry is not found; library does not exist on the specified unit; or the system source get routine ($$SYSG)[1] cannot be found. The 'mask' is a byte with one bit on and seven bits off; 'displacement' is the displacement from the value in XR1 of a 1-byte field. If SUBR15 detects the above errors, the bit position indicated by the mask is set on in the byte specified by the displacement.
6	DC DC DC	XL1'00' XL1'mask' XL1 'displacement'	Indicates that the request for service is complete. When this condition exists, the assembler program can request another entry or entries. The use of 'mask' and 'displacement' is the same as for parameter 5. (You may want to use the same displacement with a different mask.)
7	DC DC DC	XL1'00' XL1'mask' XL1 'displacement'	Indicates that a record has been passed to the assembler program from SUBR15. The use of 'mask' and 'displacement' is the same as for parameter 5. (You may want to use the same displacement with a different mask.)

Note: SUBR15 sets off the bits indicated in parameters 5, 6, and 7 each time it receives control from the user program.

ERROR IDENTIFICATION

If an error is detected, SUBR15 sets all three RPG II indicators or assembler bits off. Any of the following errors can be detected:

- The unit specified is not R1, F1, R2, or F2.
- The unit specified is not supported on the system.
- The pack on the unit specified is not initialized.
- The library specified is not S, P, O, or R.
- The record length specified is not 80 or 96 bytes.
- The unit, library, or name was changed before end-of-request was returned.
- The three RPG II indicators or assembler bits are not unique.

After one of these errors has been detected, or after a no-find error, a request can be made to SUBR15 for another library entry.

[1] $$SYSG is a system module (O-module) that is part of the SCP (system control programming) and is required to be on the system pack from which an IPL is performed. $$SYSG is not linked into the user's object program.

Appendix E. Transaction Logging—$TRLOG

Transaction logging provides the ability for batch and CCP programs to log data to a 9-track, 1600 bpi magnetic tape. This gives the user a convenient way to create an audit trail, log transactions, save program use statistics, save terminal use statistics, and save debug information.

Transaction logging consists of the following system support and assembler subroutines.

$TRLOG: This SCP program is loaded by the user into partition one and attached to the supervisor to accept information from the user program and write the tape records.

SUBR82: This SCP subroutine is supplied for use by user-written RPG II programs.

SUBR81: This SCP subroutine is supplied for use by user-written COBOL and FORTRAN programs.

In each case, the user program defines the information to be logged.

Note that transaction logging provides only the means to log transaction data to tape; it is the user's responsibility to access the tape or to otherwise use or analyze the collected information.

Using Transaction Logging

When transaction logging is used, data is passed to the supervisor ($TRLOG) by means of the subroutine (SUBR81 or SUBR82). The supervisor ($TRLOG) blocks the data into variable length blocks with a maximum record length of 2040 bytes and a maximum block length of 2048 bytes. When the block is full, it is written to tape. The user's program does not need to define a tape file for logging transactions.

Tape Considerations

The file created by $TRLOG is a single volume file with standard labels. The file characteristics are:

```
NAME-$TRLOG,REEL-$TRLOG,RECL-2044,
BLKL-2048,RECFM-VB
```

Loading $TRLOG

The transaction logging routine, $TRLOG, must be loaded into partition 1, and the start address of partition 1 must be in the first 64K of main storage. $TRLOG requires 6K of main storage, which is permanently allocated as part of the supervisor. This 6K of main storage is taken away from partition 1 and cannot be reused until the next IPL of the system. A minimum partition size of 14K is therefore required to start $TRLOG; the 6K is removed from the partition, leaving a minimum P1 size of 8K.

The OCL for $TRLOG must include a FILE statement for tape. The following OCL statements are required to load the $TRLOG program:

```
// LOAD $TRLOG,unit
// FILE NAME-$TRLOG,REEL-$TRLOG,UNIT-tape
// RUN
```

where unit is a 5444 unit code (F1, R1, F2, R2), and tape is a tape drive unit code (T1, T2, T3, T4).

When $TRLOG has been loaded, partition 1 goes to EJ. After that, the partition can be used for other user applications.

When there is a need to run $TRLOG with the System Measurement Facility (S3DSMF) and/or the system trace facility ($TRACE) the loading sequence is important. These 3 programs take their space from P1, and they must all be loaded in P1.

1. $TRLOG must be loaded first.

2. S3DSMF must be loaded next.

3. $TRACE must be loaded last.

The supervisor, with or without spool support, $TRLOG, S3DSMF, and $TRACE must all reside within the first 64K of main storage.

Controlling $TRLOG

Once $TRLOG is loaded, transaction logging is controlled through use of operator control commands. TLOG ON activates logging, and TLOG OFF deactivates logging.

The following events occur for each command:

TLOG ON

- Rewind tape (position at load point).
- Check for proper tape mounted, standard label, file name $TRLOG.
- Write header label on the tape.
- Activate tape logging.
- Send message to system console indicating tape logging is active.

TLOG OFF

- Deactivate tape logging.
- Send message to system console indicating tape logging is inactive.
- Write trailer label on tape.
- Rewind and unload the tape.

Operating Considerations

Programs using transaction logging need only to invoke a subroutine and pass the data to be logged. If logging is active, the data is moved into a 2K buffer. When the buffer is full, or when there is insufficient space left for the transaction, the buffer is written to tape. If logging is inactive, the request is ignored, and the user's program continues uninterrupted.

While logging is active, a permanent tape error or end-of-reel condition causes a hard halt which indicates the error condition. The operator can retry the operation or continue without logging. In either case, if extended restart has been generated, the system will automatically restart all programs that have halted due to the error.

The retry causes the user program to check once again to see if logging is active. If active, logging is done. If not active, logging is bypassed and execution continues.

When the option is selected to continue without logging, no further attempts to log will be made by that user program until that program has been reloaded. This means that no other logging requests will be honored, even if new and/or different transactions occur that request additional logging.

Programming Considerations

The subroutines (SUBR81 or SUBR82) are included in the user's source program during compilation.

The RPG II interface is device SPECIAL with label SUBR82 on the file description specifications. Output specifications are like any other sequential output.

The COBOL interface is:

```
CALL SUBR81 USING nnnnnn dddddd
```

where nnnnnn is defined as PICTURE 99 COMP 4 VALUE aaaa (length of the message), and dddddd is defined as PICTURE X (aaaa).

The FORTRAN interface is:

```
CALL SUBR81 (nnnnnn,dddddd)
```

where nnnnnn is a 2 byte integer previously set equal to the number of bytes in the message. The variable dddddd is an array of information at least nnnnnn bytes long.

There are two 2K buffers in $TRLOG. As one buffer fills, it is written to tape while the other buffer is being filled. Therefore, a buffer is always available to the user program. These buffers are not specified in the user program.

Additional information about transaction logging is contained in the RPG II, COBOL, and FORTRAN reference manuals.

Appendix F. Program Reference Information

System Control Programming, 5704-SC2, is shipped from the Program Library and includes many individual programs. The following is a list of these programs along with the primary source of information for that program. Most of the programs intended for customer use are described in Part 4 of this publication.

Program Name	Program	Primary References (Note 2)
The following programs are invoked with a // LOAD OCL statement:		
$$DISK	Disk Rebuild Program	SCP Data Areas Handbook
$$RSTR	Checkpoint Restart	SCP Operator's Guide; SCP Reference
$@MRGN	MRJE/WS Generation	MRJE Reference
$@MRPT	MRJE/WS Print Utility	MRJE Reference
$ALT	Alternate Track Assignment	SCP Reference
$BUILD	Alternate Track Rebuild	SCP Reference
$CE...	CE Diagnostics	CE Maintenance Manual
$CNFIG	Configuration Record Program	SCP Reference
$COPY	Copy/Dump	SCP Reference
$CPRNT	$SYSDUMP Print Program	SCP Operator's Guide
$DCOPY	Dump/Restore	SCP Reference
$DELET	File Delete	SCP Reference
$DISK@	Disk Address Compare Program	SCP Data Areas Handbook
$DUMP	Dump Tape and Disk	SCP Data Areas Handbook
$FCOMP	File Compress	SCP Reference
$HIST	System History Display	SCP Reference
$INIT	Initialize Disk	SCP Reference
$KLEAN	Chain Cleaning Program	SCP Reference
$LABEL	File and Volume Label Display	SCP Reference
$MAINT	Library Maintenance	SCP Reference
$MPXDV	Macroprocessor	Macros Reference
$OLINK	Overlay Linkage Editor	OLE Reference
$QCOPY	Spool File Copy	SCP Reference
$RINDX	Recover Index	SCP Reference
$RSALT	Reassign Alternate Track	SCP Reference
$SCOPY	Simulation Area Program	SCP Reference
$SG... (Note 1)	System Generation	System Generation
$SGFIX	Program Temporary Patch	SCP Data Areas Handbook
$SGLOG	PTF List	SCP Data Areas Handbook
$SGPTF	Program Temporary Fix	SCP Data Areas Handbook
$SGPTR	Program Temporary Fix	SCP Data Areas Handbook
$SGPVR	Program Temporary Fix	SCP Data Areas Handbook
$TINIT	Initialize Tape	SCP Reference
$TRACE	Trace Routine	SCP Data Areas Handbook
$TRLOG	Transaction Logging	SCP Reference
$TRPRT	Print Trace Entries	SCP Data Areas Handbook
$TVES	Tape Error Summary	SCP Reference
$WVTOC	VTOC Conversion	SCP Reference
The following program components are distributed with SCP 5704-SC2 but are not loaded by means of a // LOAD OCL statement:		
SUBR07	MICR-1255	MICR Reference
SUBR08	MICR-1255	MICR Reference
SUBR09	MICR-1419	MICR Reference
SUBR15	Library Entry Retrieval	SCP Reference
SUBR81	Transaction Logging Subroutine	SCP Reference
SUBR82	Transaction Logging Subroutine	SCP Reference
	Operator Control Commands (OCC)	SCP Operator's Guide
	Spool Commands	Spool User's Guide
	I/O Macros	Macros Reference
	ML/MP Macros	MLMP Reference

Note 1: $SG. . . refers to system generation functions excluding those $SG programs specifically listed.

Note 2: Legend

SCP Operator's Guide	*IBM System/3 Model 15 Operator's Guide,* GC21-5075
SCP Data Areas Handbook	*IBM System/3 Model 15 System Data Areas and Diagnostic Aids,* SY21-0052. (The programs described in this publication are intended for use by the CE program support representative.)
Spool User's Guide	*IBM System/3 Model 15 User's Guide to Spooling,* GC21-7632
OLE Reference	*IBM System/3 Overlay Linkage Editor Reference Manual,* GC21-7561
Macros Reference	*IBM System/3 Model 15 System Control Programming Macros Reference Manual,* GC21-7608
System Generation	*IBM System/3 Model 15 System Generation Reference Manual,* GC21-7616
MLMP Reference	*IBM System/3 Multiline/Multipoint Binary Synchronous Communications Reference Manual,* GC21-7573
MRJE Reference	*IBM System/3 Model 15 MULTILEAVING Remote Job Entry/ Work Station Support Reference Manual,* GC21-5115
MICR Reference	*IBM System/3 Models 8, 10, 12, and 15 1255 and 1419 Magnetic Character Readers Reference and Program Logic Manual,* GC21-5132
CE Maintenance Manual	*MDM — Volume 1A CE Diagnostic User Guide* (which is sent with the system)

Index

// CEND 4-129
// NEND 4-129
/. statement 1-20, 1-96
/. statement (OCL) 1-13
/@ statement 1-20, 1-95
/@ statement (OCL) 1-12
$ALT (alternate track assignment program) 4-7
$BUILD (alternate track rebuild program) 4-12
$CNFIG
  ASSIGN statement 1-21
  during system generation 3-9
$CNFIG (configuration record program) 4-17
$COPY (copy/dump program) 4-27
$DCOPY (dump/restore program) 4-62
$DELET (file delete program) 4-71
$FCOMP (file compress program) 4-78
$HACCP (system history area copy program) 4-93
$HIST (system history area display program) 4-93
$INDEX40 4-209
$INDEX45 2-24, 4-209
$INIT (disk initialization program) 4-100
$KLEAN (chain cleaning program) 4-108
$LABEL (file and volume label display program) 4-109
$MAINT (library maintenance program) 4-117
$QCOPY (spool file copy program) 4-167
$QCOPY files and subroutines 4-185
$RINDX (recover index program) 4-208
$RSALT (reassign alternate track program) 4-212
$SCOPY (simulation area program) 4-216
$SPOOL file 4-78
$TINIT (tape initialization program) 4-233
$TVES (tape error summary program) 4-239
$WVTOC (VTOC service program) 4-241
* (COMMENT) statement 1-99
* (COMMENT) statement (OCL) 1-13
* parameter for the LOAD statement (OCL) 1-18
ACCESS statement ($COPY)
  CYLINDER parameter 4-33
  DISP parameter 4-33
  FROM parameter 4-33
  RECL parameter 4-33
  SECTOR parameter 4-33
  TRACK parameter 4-33
accessing added records 2-70
accessing simulation areas 3-7
add files from one main data area to another 4-78
adding a missing parameter to a procedure 2-33
adding a statement to a procedure 2-33
adding source library entries 4-137
additional disk identification 4-102
advantages of nested procedures 2-35
AFTER parameter 4-152
allocate considerations and restrictions 4-119
allocate disk space 3-11
ALLOCATE statement ($MAINT)
  allocation of disk work space 4-123
  control statement summary 4-118
  DIRSIZE parameter 4-121
  function 4-118
  HISTORY parameter 4-122
  OBJECT parameter 4-126
  PACKO parameter 4-123
  parameter summary 4-120
  restrictions 4-119
  SOURCE parameter 4-124
  SYSTEM parameter 4-121
  TO parameter 4-121
  WORK parameter 4-122
allocate work space for keysort 2-23
allocation limit 4-121
ALT statement ($ALT)
  ASSIGN parameter 4-7
  control statement summary 4-7
  PACK parameter 4-7
  parameter summary 4-8
  UNIT parameter 4-7
  VERIFY parameter 4-7
alternate track assignment program
  ALT statement 4-7
  conditional assignment 4-9
  examples 4-10
  messages 4-11
  OCL considerations 4-10
  unconditional assignment 4-9

alternate track rebuild program
examples 4-15
OCL considerations 4-15
REBUILD statement 4-12
substitute data 4-14
alternate tracks
definition of 3-11
file processing considerations 3-11
location 3-11
appendix A A-1
appendix B B-1
appendix C C-1
area codes
main data 3-14, 4-6, 4-218
simulation 1-14, 1-21, 4-6
ASCII parameter, FILE statement 1-51
assign a class number to a program
considerations and restrictions 4-196.3
control statement 4-170
parameter description 4-184
parameter summary 4-174.2
ASSIGN parameter ($ALT) 4-7
ASSIGN statement 1-14
considerations and restrictions 1-22
contents 1-21
example 1-21
format 1-21
function 1-21
OCL 1-9
parameters 1-14
F1 1-21
F2 1-21
R1 1-21
R2 1-21
placement 1-21
assigning simulation areas 3-7
assigning 5444 unit codes 3-7
assignment of alternate tracks
alternate track assignment program 4-7, 4-9
disk initialization program 4-104
assignment of simulation areas 3-7
asterisk parameter (use in LOAD OCL statement) 1-75
automatic file allocation 2-13, 2-23
automatic message restart 2-70
available simulation areas 3-7

backup function 4-79
basic assembler 2-22
batch processing 2-56
BLKL parameter, FILE statement 1-49
braces C-1, 4-5
brackets C-1, 4-5

BSCA statement 1-9
contents 1-23
format 1-23
function 1-23
parameters 1-14
parameters, line 1-23
placement 1-23
spooling considerations 1-23
build a direct file 4-27
building a new index 4-35

calculating file size
converting cylinder/track to track number B-7
converting track number of cylinder track B-8
main data area B-7
number of tracks
in a sequential file B-7
in an indexed file B-7
of disk track index B-7
CALL statement
contents 1-24
example 1-24
format 1-24
function 1-24
parameters 1-14
procedure name 1-14
switch characters 1-14
unit 1-14
placement 1-24
spooling consideration 1-24
CALL statement, OCL 1-9
CANCEL command C-2
CANCEL OCC command C-2
capital letters, numbers, and special characters 1-5, 4-5
capitalized words and letters 1-5, 4-5
cataloging to an active library 2-39
CCP assignment set 4-204
CEND 4-129
chain cleaning program 4-108
chain-image area
changing 4-108
entering from
source library 1-67
system input device 1-67
CHANGE command C-2.1
CHANGE OCC command C-2.1
change the support for cataloging to a program pack 4-117
changing a partition 1-72.1
changing a temporary file to a scratch file 1-34
changing procedure parameters 2-33
changing punch device (see PUNCH statement)
changing simulation area code assignments 3-9

changing the configuration record 4-17
changing the contents of the chain-image area 1-66
changing the log device 1-79
changing the name of a library entry 4-153
changing the number of lines the printer will print per page 1-85
changing the size of the object library 4-127
changing the size of the source library 4-124
changing the system input device 1-91
changing 5444 unit code assignments 3-9
CHAR format parameter for the IMAGE statement 1-67
characters from the source library on disk 1-67
characters from the system input device 1-67
characters from the system input device, example 1-68
characters to use when naming library entries 4-135
checkpoint/restart 2-68
choosing the designation of a library entry
  permanent 4-136
  temporary 4-136
classify batch programs (see *assign a class number to a program*)
cleaning the print chain 4-108
clear initialization 4-100
CLOSE parameter 1-19
COBOL 2-21
codes for main data areas 3-14, 4-6
codes for simulation areas 3-14, 4-6
coding rules
  /. 1-5
  /& 1-5
  * 1-7
  blanks 1-5
  braces 1-5
  brackets 1-5
  capitalized words 1-5
  commas 1-5
  continuation 1-6
  control statements 4-4
  hyphen 1-4
  keyword parameters 1-4
  numbers 1-5
  OCL
    comment 1-7
    continuation 1-6
    statements beginning with // 1-5
    statements beginning with other than // 1-5
  positional parameters 1-5
  special characters 1-5
  underscore 1-5
combinations of seven track specifications 1-54
comments 1-7
  coding rules 1-7
  comment statement 1-99
  jobname 1-7
  stepname 1-7
compatible disk access methods 2-18.2
compilation date 4-134
COMPILE statement 1-25
  contents 1-25
  example 1-26
  format 1-25
  function 1-25
  OCL 1-9
  parameters 1-14
    ATTR 1-26
    LINKADD 1-25
    OBJECT 1-25
    SOURCE 1-25
    UNIT 1-25
  placement 1-25
  spooling considerations 1-26
compiling a source program
  storing it in object library 2-30
  storing it in object library sample statement 2-31
compress files high 4-78
compress files low 4-78
compress in place 4-127
compressing files 4-78, 4-79
conditional assignment of alternate tracks
  example 4-10
  incorrect data 4-9
  surface analysis 4-9
configuration record program 4-17
CONSOLE parameter
  LOG statement 1-81
  READER statement 1-91
continuation (OCL) 1-6
continuation, example 1-6
control statement summary (system service programs)
  ACCESS statement ($COPY) 4-30
  ALLOCATE statement ($MAINT) 4-118
  ALT statement ($ALT) 4-7
  ALTA statement ($RSALT) 4-212
  ASNP statement ($CNFIG) 4-18
  AUTHORIZE statement ($QCOPY) 4-170
  AUTST statement ($CNFIG) 4-18
  AUTWT statement ($CNFIG) 4-18
  BLANK statement($CNFIG) 4-18
  CATLG statement ($CNFIG) 4-18
  CEND statement ($MAINT) 4-129
  CLASSIFY statement ($QCOPY) 4-170
  CLEAR statement ($SCOPY) 4-217
  COMPRESS statement ($WVTOC) 4-241
  CONSOL statement ($CNFIG) 4-18
  COPY statement ($MAINT) 4-129
  COPYAREA statement ($SCOPY) 4-217
  COPYCTRL statement ($QCOPY) 4-169
  COPYFILE statement ($COPY) 4-29

control statement summary (system service programs) (continued)
COPYFILES statement ($FCOMP) 4-80
COPYIPL statement ($SCOPY) 4-217
COPYPACK statement ($COPY) 4-29
COPYPACK statement ($DCOPY) 4-62
COPYPCHQ statement ($QCOPY) 4-168
COPYPRTQ statement ($QCOPY) 4-168
COPYQ statement ($QCOPY) 4-170
COPYRDRQ statement ($QCOPY) 4-169
COPYSP statement ($QCOPY) 4-168
DEFCN statement ($CNFIG) 4-18
DEFFN statement ($CNFIG) 4-18
DELETE statement ($MAINT) 4-146
DISPLAY statement ($LABEL) 4-110
DISPLAY statement ($QCOPY) 4-169
FORMAT statement ($CNFIG) 4-18
FORMAT statement ($DELET) 4-72
FSHARE statement ($CNFIG) 4-18
HLTP statement ($CNFIG) 4-18
INSERT statement ($MAINT) 4-150
ITYPE statement ($CNFIG) 4-18
LOGP statement ($CNFIG) 4-18.1
MESSAG statement ($CNFIG) 4-18.1
MODIFY statement ($MAINT) 4-150
MOVE statement ($SCOPY) 4-217
NAMES statement ($SCOPY) 4-217
NEND statement ($MAINT) 4-129
NEWNAME statement ($SCOPY) 4-217
OUTDM statement ($COPY) 4-30
OUTPUT statement ($HIST) 4-94
PRINT statement ($CNFIG) 4-18.1
PRINT statement ($HIST) 4-94
PRIORITY statement ($CNFIG) 4-18.1
QCOPY statement ($CNFIG) 4-18.1
READY statement ($CNFIG) 4-18.1
REBUILD statement ($BUILD) 4-12
REMOVE statement ($DELET) 4-72
REMOVE statement ($MAINT) 4-150
RENAME statement ($MAINT) 4-153
REPLACE statement ($MAINT) 4-150
RESTORE statement ($QCOPY) 4-169
SELECT statement ($COPY) 4-29
SHA statement ($CNFIG) 4-18.1
SIZE statement ($CNFIG) 4-18.1
SPCYL statement ($CNFIG) 4-18.1
SPDSK statement ($CNFIG) 4-18.1
SPEXT statement ($CNFIG) 4-18.2
SPOPT statement ($CNFIG) 4-18.1
SYIN statement ($CNFIG) 4-18.2
SYPC statement ($CNFIG) 4-18.2
SYPR statement ($CNFIG) 4-18.2
TAPEFILES statement ($FCOMP) 4-80
TSTAMP statement ($CNFIG) 4-18.2
UIN statement ($INIT) 4-101
VOL statement ($INIT) 4-101
VOL statement ($TINIT) 4-234
controlling $TRLOG E-2
conversion
convert seven track tape 1-51
cylinder-tracks B-8
records-tracks B-7
track number-cylinder number B-7
copy a display of the status of the spool queues
considerations and restrictions 4-194
control statement 4-169
parameter description 4-181
parameter summary 4-174
copy an index 4-35
copy/dump program
card input considerations 4-40
card output considerations 4-41
COPYFILE statement (see COPYFILE statement)
copying multivolume files (see copying multivolume files)
COPYPACK statement (see COPYPACK statement)
diskette (3741) file consideration 4-39
examples 4-42
OCL considerations 4-41
SELECT statement (see SELECT statement)
tape file considerations 4-39
use of 2-15
copy jobs to or from the reader queue
considerations and restrictions 4-192
control statement 4-169
parameter description 4-178
parameter summary 4-173
copy selected job steps from one spool file to another 4-195
considerations and restrictions 4-195
control statement 4-170
parameter description 4-183
parameter summary 4-174.1
copy selected jobsteps from the print queue
considerations and restrictions 4-184
control statement 4-168
parameter description 4-176
parameter summary 4-171
copy selected jobsteps from the punch queue
considerations and restrictions 4-190
control statement 4-168
parameter description 4-177
parameter summary 4-172
COPY statement ($MAINT)
control statement summary 4-129
file-to-library 4-129
FROM parameter 4-132
function 4-128
LIBRARY parameter 4-132
library-to-card 4-131
library-to-library 4-130
library-to-printer/card consideration 4-139
NAME parameter 4-132

NEWNAME parameter 4-134
PACKIN parameter 4-134
PACKO parameter 4-134
parameters 4-132
reader-to-library 4-129
RETAIN parameter 4-133
TO parameter 4-133
uses 4-128
copy the entire spool file
considerations and restrictions 4-184
control statement 4-168
parameter description 4-176
parameter summary 4-171
COPYFILE statement ($COPY)
control statement summary 4-28
DELETE parameter 4-36
LENGTH parameter 4-36
OMIT parameter 4-36
OUTPTX parameter 4-34
OUTPUT parameter 4-34
parameter summary 4-31
REORG parameter 4-36
COPYIN 4-27, 4-33
copying an entire area 4-34
example 4-43
copying an index 4-35
copying files ($COPY)
example 4-42
disk to tape 4-45
tape to disk 4-46
copying minimum system from one disk to another 4-130
example 4-158
copying multivolume files
copying multivolume indexed files 4-38
direct file attributes 4-38
maintaining correct relative record numbers 4-38
maintaining proper volume sequence numbers 4-38
copying object program to F1, example 4-159
COPYO 4-27
COPYPACK considerations 4-34
COPYPACK statement ($COPY)
control statement summary 4-29
FROM parameter 4-31
PACKIN parameter 4-34
PACKO parameter 4-34
parameter summary 4-31
TO parameter 4-31
COPYPACK statement ($DCOPY)
BACKUP parameter 4-64
control statement summary 4-62
FROM parameter 4-63
PACK parameter 4-64
parameter summary 4-63
SYSTEM parameter 4-64
TO parameter 4-63
core index B-6
CORE parameter for JOB statement 1-72
correcting characters on an alternate track 4-12
correcting characters on an alternate track, example 4-15
correcting index problems 4-35
create a new index 4-35
creating a source library 4-124
creating an object library 4-126
creating disk files 2-12
cylinder assignment 3-6
cylinder 0 format 3-12

data area track requirements B-1
DATA parameter for the REMOVE statement ($DELET) 4-72
DATE command C-2.1
DATE OCC command C-2.1
date of compilation 4-134
DATE parameter
FILE statement (OCL)
disk 1-35
tape 1-49
REMOVE statement ($DELET) 4-72
DATE parameter for the DATE statement (OCL) 1-27
DATE statement
contents 1-27
example 1-28
format 1-27
function 1-27
OCL 1-9
parameters 1-14
placement 1-27
spooling considerations 1-28
date support
job date 2-63
jobstep date 2-63
partition date 2-63
system date 2-63
default simulation area assignments 3-9
DEFER parameter, FILE statement 1-46
definition
external buffers 2-9
main data area 3-10
object module 2-4
simulation areas 3-6
source programs 2-4
volume 3-3
DELETE parameter for the COPYFILE statement ($COPY) 4-36
DELETE statement ($MAINT)
control statement summary 4-146
FROM parameter 4-148
function 4-146
LIBRARY parameter 4-148
NAME parameter 4-148

delete statement ($MAINT) (continued)
PACKIN parameter 4-148
parameters 4-148
restrictions 4-147
RETAIN parameter 4-148
uses 4-146
deleting a procedure parameter 2-33
deleting a source library 4-125
deleting an object library 4-127
deleting library entries ($MAINT)
entries of one type 4-146
entries of one type, example 4-163
entries with names beginning with certain characters 4-146
example 4-162
entry from a library 4-146
entry from a library, example 4-162
library entries 4-146
temporary or permanent entries 4-146
certain type 4-146
contain characters 4-146
same name 4-146
deleting one of several files having the same name ($DELET) 4-75
DENSITY parameter
FILE statement 1-50
VOL statement 4-235
direct access storage 3-3
direct file attributes 4-38
direct file tracks B-7
DIRSIZE parameter for ALLOCATE statement ($MAINT) 4-120, 4-121
disk capacity 3-5
disk data management 2-3
disk FILE statement 1-29
disk initialization program
clear initialization 4-100
CYL0 initialization 4-101
example 4-106
force initialization 4-101
messages 4-107
OCL considerations 4-106
primary initialization 4-101
rename initialization 4-101
UIN statement (see UIN statement)
VOL statement (see VOL statement)
disk requirements for data records B-2
disk sort 2-22
disk space allocation
scratch files 3-10
temporary or permanent files 3-10
disk storage 3-1
disk storage configurations 3-3
diskette (3741) file considerations 4-39
DISP parameter for the REBUILD statement 4-14
DISPLAY command C-2.2
DISPLAY OCC command C-2.2
DISPLAY statement ($LABEL)
control statement summary 4-110
LABEL parameter 4-110
parameter descriptions 4-111
parameter summary 4-111
SORT-NAME parameter 4-111
UNIT parameter 4-110
displaying the spool queues
considerations and restrictions 4-194
control statement 4-169
parameter description 4-181
parameter summary 4-174
doubly-defined files 2-18
DTF (define the file) 2-17
DUMP command C-2.2
DUMP OCC command C-2.2
dump/restore program ($DCOPY)
COPYPACK statement 4-63
FROM parameter 4-63
OCL considerations 4-65
TO parameter 4-63
dumping disk to tape 4-62

END control statement 4-5
end-of-data (see /* statement)
end-of-data pointer 4-208
end-of-index pointer 4-208
END parameter, FILE statement 1-50
END statement 4-5
entering OCC commands C-1
entries with the same name 4-135
ERASE parameter for the UIN statement 4-104
extended restart 2-70
external buffer support
basic assembler 2-10
COBOL 2-9
disk sort 2-10
FORTRAN 2-9
RPG II 2-9
external buffers 2-9
definition of 2-9
use of 2-9
external indicators 1-94, 2-48

file and volume label display program ($LABEL)
DISPLAY statement 4-110
examples 4-116
meaning of VTOC information 4-113
OCL considerations 4-111
VTOC printout 4-116

file compress program ($FCOMP) 4-78
  considerations and restrictions 4-83
  control statement summary 4-80
  examples 4-84
  move and copy function
    copy function 4-78
    move function 4-78
  OCL considerations 4-84
  parameter summary 4-80
  program description 4-78
file delete program
  examples 4-75
  OCL considerations 4-75
  REMOVE statement 4-72
file facilities 2-11
file index tracks B-5
file location on main data area 3-10
file names 1-30, 1-46
file names used in the FILE statement 1-30, 1-46
file recovery
  control statement 4-30
  main data area 4-30
  simulation area 4-30
file share area 2-18
file sharing 2-16, 2-57
FILE statement (disk)
  parameters
    contents 1-29
    DATE 1-29
    examples 1-36
    file names 1-30
    file processing considerations 1-39
    format 1-29
    function 1-29
    HIKEY 1-43
    LABEL 1-29
    LOCATION 1-29
    NAME 1-29
    PACK 1-29
    placement 1-29
    programs requiring specific file names 1-30
    RECORDS 1-29
    RETAIN 1-29
    SHARE 1-29
    specific file names 1-30
    spooling considerations 1-39
    TRACKS 1-29
    UNIT 1-29
    VERIFY 1-29
FILE statement (disk), programs and file names 1-30
FILE statement (multifile tape volumes)
  contents 1-57
  preposition tapes 1-58
  REEL parameter 1-62
  restrictions on use 1-58
  SEQNUM parameter 1-57
  spooling considerations 1-62
  standard labeled files 1-59
  unlabeled volumes 1-61
FILE statement (multivolume disk files)
  contents 1-40
  parameters
    HIKEY 1-43
    LOCATION 1-43
    PACK 1-41
    RECORDS 1-42
    RETAIN 1-43
    TRACKS 1-42
    UNIT 1-41
FILE statement (multivolume tape files)
  contents 1-55
  examples 1-56
  parameters
    REEL 1-55
    UNIT 1-55
  spooling considerations 1-56
FILE statement (OCL)
  example 1-36
  file processing considerations 1-39
  FILE statement considerations (backup and restore) 4-83
  FILE statement considerations for multivolume files 1-55
  format
    disk 1-29
    tape 1-45
  function
    disk 1-29
    tape 1-45
  parameter
    ASCII 1-51
    BLKL 1-49
    CONVERT 1-51
    DATE, disk 1-35
    DATE, tape 1-49
    DEFER 1-51
    DENSITY 1-50
    END 1-50
    HIKEY 1-43
    HIKEY (packed) 1-44
    LABEL, disk 1-32
    LABEL, tape 1-48
    LOCATION 1-33
    NAME, disk 1-29
    NAME, tape 1-46
    PACK 1-31
    PARITY 1-51
    RECFM 1-50
    RECL 1-50
    REEL 1-48
    RETAIN, disk 1-34
    RETAIN, tape 1-49
    SEQNUM 1-52
    SHARE 1-36
    TRACKS or RECORDS 1-32
    TRANSLATE 1-51
    UNIT, disk 1-31
    UNIT, tape 1-47
    VERIFY 1-36

file statement (OCL) (continued)
placement
disk 1-29
tape 1-45
FILE statement (single volume tape files)
contents 1-45
format 1-45
function 1-45
parameter
ASCII 1-45
BLKL 1-45
CONVERT 1-46
DATE 1-45
DEFER 1-46
DENSITY 1-45
END 1-45
LABEL 1-45
NAME 1-45
PARITY 1-46
RECFM 1-45
RECL 1-45
REEL 1-45
RETAIN 1-45
SEQNUM 1-46
TRANSLATE 1-46
UNIT 1-47
placement 1-45
FILE statement (tape)
combinations of seven track specifications 1-54
files names 1-46
parameter applicability for nine track 1-53
parameter applicability for seven track 1-53
programs and file names 1-46
programs requiring specific file names 1-46
seven track considerations 1-52
specific file names 1-46
tape file statement summary 1-53
FILE statement parameters
device independent 1-17
disk 1-15
tape 1-16
file-to-library copy function of library maintenance program 4-129
files
automatic allocation 2-13
creation 2-12
definition 2-11
disk files 2-12
large indexed files 2-23
location 2-13
multifile tape volumes 2-25
multivolume disk files 2-24
multivolume tape files 2-25
organization
direct 2-11
indexed 2-11
sequential 2-11
files (continued)
processing
consecutive 2-11
random 2-11
sequential 2-11
programming considerations 2-26
SDTF (share define the file) 2-17
services 2-15
sharing files
compatible access methods 2-17
compatible disk access methods 2-18.2
considerations and restrictions 2-18.1
doubly-defined files 2-18
DTF (define the file) 2-17
file share area 2-18
FSQE (file share queue element) 2-18
work files 2-20
files and subroutines (QCOPY) 4-185
format of cylinder 0 3-12
format of data module 3-3
format of OCL statement
/. statement 1-96
/ & statement 1-95
/* statement 1-100
* statement 1-99
ASSIGN statement 1-21
BSCA statement 1-23
CALL statement 1-24
COMPILE statement 1-25
DATE statement 1-27
FILE statement
device independent files 1-63
multifile tape volumes 1-57
multivolume disk files 1-40
multivolume tape files 1-55
single volume disk files 1-29
single volume tape files 1-45
HALT statement 1-65
IMAGE statement 1-66
JOB statement 1-71
LOAD and LOAD* statement 1-75
LOG statement 1-80
NOHALT statement 1-82
PAUSE statement 1-84
PRINTER statement 1-85
PUNCH statement 1-88
READER statement 1-91
RUN statement 1-92
SWITCH statement 1-93
format of volume 3-3
FORMAT parameter for the IMAGE statement
CHAR 1-66
HEX 1-66
MEM 1-66
FORMAT statement for the file delete program
example 4-77
purpose 4-72
FORTRAN 2-21
free up allocated, but unused space 4-71

free up allocated, but unused space, example 4-77
FROM parameter
ACCESS statement ($COPY) 4-33
CLEAR statement ($SCOPY) 4-222
COPY statement ($MAINT) 4-132
COPYAREA statement ($SCOPY) 4-219
COPYFILES statement ($FCOMP) 4-80
COPYIPL statement($SCOPY) 4-223
COPYPACK statement ($COPY) 4-31
COPYPACK statement ($DCOPY) 4-63
COPYSP statement ($QCOPY) 4-176
DELETE statement ($MAINT) 4-148
MODIFY statement ($MAINT) 4-151
MOVE statement ($SCOPY) 4-224
RENAME statement ($MAINT) 4-153
SELECT KEY statement ($COPY) 4-36
SELECT RECORD statement ($COPY) 4-37
TAPEFILES statement ($FCOMP) 4-81
FSQE (file share queue element) 2-18
function of OCL statements (see desired statement type)
function of system service programs 4-3

gaps in the object library 4-127
general coding rules 1-4
general coding rules (see coding rules)
general information, model 15D xi
greater than 48K programs 2-9

HALT command C-3
HALT OCC command C-3
HALT statement 1-17, 1-65
HALT statement (OCL) 1-10
halt 90 1-84
halt 91 1-84
HEX format parameter for the IMAGE statement 1-66
HIKEY parameter for the FILE statement 1-43
HIKEY parameter for the FILE statement, packed 1-44
HISTORY parameter, ALLOCATE statement ($MAINT) 4-122
HOLD command C-3
how to change the configuration record 4-17
how to request $QCOPY from a terminal 4-205
how to use this manual ix
how to write control statements 4-4

IBM System/3 standard character set A-1
ID parameter, VOL statement ($TINIT) 4-235
IDELETE command C-3
IMAGE statement 1-17
IMAGE statement (OCL) 1-10
IMAGE statement (OCL), spooling considerations for 1-68
IMAGE statement parameters 1-17
INCLUDE statement 1-17
example 1-70
function 1-69
OCL 1-10
table of parameters 1-17
including comments in OCL statement 1-7
including system programs in a library 4-121
incorrect data 4-12, 4-16
INCR parameter of MODIFY statement 4-151
index area track requirements B-4
index problems 4-35
indexed file tracks B-5
INDEX40 2-24, 4-209
INDEX45 2-24, 4-209
indicate the number of lines per page the printer will print 1-85
indicator-settings parameter for the SWITCH statement 1-93
initial program load (IPL) 2-42, 3-13
initializing disks 4-100
initializing tapes 4-233
input device, changing (see READER statement)
input file name 4-27
inquiry program 2-69
INSERT statement ($MAINT)
control statement summary 4-150
functions 4-149
parameters 4-152
inserting source library statements 4-149
interchanging data modules 3-6
interval timer 2-63
introduction xi
introduction to OCL statements 1-3
introduction-what is OCL 1-3
IPL (initial program load) 2-42, 3-13

job and job step classification
with nested procedures 2-47
without procedures 2-46
job and jobstep processing 2-44
job mode 2-45
job processing example 2-46
JOB statement 1-71
JOB statement (OCL) 1-10
JOB statement parameters 1-18
job stream 2-48
job stream, delimiters 1-96

job stream, relationship to OCL 1-3
job stream, sample 1-3
jobname in JOB OCL statement 1-71
jobstep mode 2-44

KEEP command OC-4
KEEP OCC command OC-4
keysort workspace 2-24
keyword 1-4
keyword parameter 1-3

LABEL parameter
  DISPLAY statement 4-110
  FILE statement
    disk 1-32
    tape 1-45
  REMOVE statement 4-72
labeled tapes 2-26
length of names given to library entries 4-135
LENGTH parameter
  COPYFILE statement 4-36
  REBUILD statement 4-14
libraries
  cataloging to active program packs 2-39
    all program packs 2-39
    CCP program packs 2-39
    no program packs 2-39
    user considerations 2-41
  definition 2-26.1
  location 2-29
  object 2-27
  procedures 2-32
  nested 2-35
  source 2-26.1
library directory printouts 4-141
  object library 4-142
  source library 4-141
  system directory 4-145
library entries, removing temporary 4-119, 4-136
library entry retrieval subroutine D-1
library facilities 2-26.1
library maintenance
  allocate considerations and restrictions 4-119
  copy considerations and restrictions 4-131
  delete considerations and restrictions 4-147
  modify considerations and restrictions 4-149
  rename considerations and restrictions 4-153
library maintenance program
  ALLOCATE statement (see ALLOCATE statement)
  COPY statement (see COPY statement)
  DELETE statement (see DELETE statement)
  examples 4-156
  library description, use of disk space 4-117
  MODIFY statement 4-150
  multiprogramming considerations and restrictions 4-119
  OCL considerations 4-155
  RENAME statement (see RENAME statement)
LIBRARY parameter
  COPY statement 4-129
  DELETE statement 4-146
  MODIFY statement 4-150
  RENAME statement 4-153
library-to-card considerations ($MAINT) 4-139
library-to-library considerations ($MAINT) 4-138
library-to-printer considerations ($MAINT) 4-139
library, object (see object library)
library, source (see source library)
linking SUBR15 with assembler D-5
linking SUBR15 with RPG II D-2
LIST parameter of MODIFY statement 4-151
listing source library statements 4-149
LOAD * programs 2-44
LOAD * statement 1-75
LOAD * statement (OCL) 1-11
load modules 2-5
LOAD statement 1-75
LOAD statement (OCL)
  * parameter 1-75
  example 1-77
  format 1-75
  function 1-75
  placement 1-75
  program-name1 parameter 1-76
  program-name2 parameter 1-76
  stepname entry 1-75
  switch characters 1-76
  unit1 parameter 1-76
  unit2 parameter 1-77
LOAD statement parameters 1-18
loading and running programs 2-43
loading programs from a file 1-75
loading programs from disk
  DATE statement 1-27
  HALT statement 1-65
  IMAGE statement 1-66
  JOB statement 1-71
  LOAD statement 1-75
  NOHALT statement 1-82
  sample job streams 2-48
loading TRLOG E-1
location of files on main data area 3-10
location of object library 2-29
location of scratch files 3-10

location of source library 2-27
location of system pack 3-14
LOCATION parameter for the FILE statement 1-43
log device 1-79
LOG statement (OCL) 1-10
LOG statement parameters 1-18

magnetic tape (see tape, magnetic)
main data area
  capacity 3-5
  codes 3-14, 4-6, 4-218
  definition of 3-10
  use of 3-10
main storage usage 2-71
maintain the AUTHORIZ file
  authorization change records 4-196
  authorization field
    position 1 4-196.1
    position 2 4-196.1
    positions 3 and 4 4-196.2
  authorization file 4-196
  change authorization record 4-196.3
  considerations and restrictions 4-196
  control statement 4-170
  create authorization record 4-196.3
  delete authorization record 4-196.3
  parameter description 4-183
  parameter summary 4-174.2
maintain the authorize file (AUTHORIZE) 4-196
maintaining correct record number (multivolume file) 4-38
maintaining volume sequence numbers (multivolume file) 4-38
maximum number of assigned simulation areas 3-9
maximum number of control statements in a procedure 2-32
maximum number of files in SWA 2-15
maximum number of levels that can be nested together 2-37
maximum number of volumes 2-24
maximum program size 2-8
meaning of VTOC information 4-112
MEM, format parameter for the IMAGE statement 1-17
message options 2-19
messages
  alternate track assignment program 4-11
  disk initialization program 4-107
  dump/restore 4-66, 4-77
  tape initialization 4-236
MFCM1 parameter
  PUNCH statement 1-88
  READER statement 1-91
MFCM2 parameter
  PUNCH statement 1-88
  READER statement 1-91
MFCU1 parameter
  PUNCH statement 1-88
  READER statement 1-91
MFCU2 parameter
  PUNCH statement 1-88
  READER statement 1-91
minimum number of assigned simulation areas 3-9
Model 15D general information xi
Model 15D introduction xi
MODIFY statement ($MAINT)
  control statement summary 4-150
  functions 4-149
  parameters 4-151
  restrictions 4-149
moving object library 4-119
multifile tape volumes 2-25
  null files on tape 2-25
    labeled tapes 2-26
    unlabeled tapes 2-25
multifile volumes, tapes 1-57
multiprogramming
  considerations and restrictions 2-57
  examples 2-60
  operation 2-52
  operator control commands 2-54
  sharing files 2-57
multivolume files
  copying 4-38
  disk 1-40
  tape 1-55
multivolume tape files 2-25

name of entry to be deleted 4-148
name of entry to be renamed 4-154
NAME parameter
  COPY statement 4-132
  DELETE statement 4-148
  FILE statement
    device independent 1-63
    disk 1-30
    tape 1-46
  IMAGE statement 1-67
  RENAME statement 4-154
NAME360 parameter, VOL statement 4-105
naming library entries
  characters to use 4-135
  length 4-135
  restrictions 4-135
NEND 4-129
nested procedure restart 2-38
nested procedures
  advantages 2-43
  examples 2-35
  maximum number of levels that can be nested 2-37
  rules 2-37

new names to be given to an entry, rules 4-135
NEWNAME parameter
COPY statement 4-134
RENAME statement 4-154
NOHALT command C-3
NOHALT statement
OCL 1-11
OCL, severity codes for 1-83
parameters 1-19
NOIDELETE command C-4
normal procedure call 2-32
null files on tape 2-25
number of alternate tracks on a disk 3-11
number of available simulation areas 3-7
number of files in SWA 2-15
number of tracks in a direct file B-7
number of tracks in a sequential file B-7
number of tracks in an indexed file B-7
NUMBER parameter for the IMAGE statement 1-67

object library
changing size 4-127
creating 4-126
deleting 4-127
gaps 2-29, 4-127
location 2-29
maintaining 2-29
organization 2-28
physical characteristics
directory 2-27
size 2-27
reorganizing 4-127
object modules 2-4
OBJECT parameter
ALLOCATE statement 4-126
COMPILE statement 1-25
OCC commands C-1
OCL
code parameter 1-4
coding rules (see coding rules)
data parameter 1-4
input job stream 1-3
introduction 1-3
job stream 1-3
length 1-6
means of supplying 1-3
parameters 1-4
special characters (see coding rules)
spooling considerations 1-7
statement
example 1-4
identifiers 1-4
OCL considerations for spooling and multiprogramming 1-7
OCL considerations for system service programs
alt track assignment program 4-10
alt track rebuild program 4-15
chain cleaning program 4-108
configuration record program 4-24.2
copy/dump program 4-41
disk initialization program 4-101
dump/restore program 4-65
file and volume label display program 4-116
file compress program 4-84
file delete program 4-75
library maintenance program 4-155
recover index program 4-210
simulation area program 4-225
spool file copy program 4-196.6
system history area display program 4-94
tape error summary program 4-240
tape initialization program 4-236
VTOC service program 4-242
OCL parameters table 1-14
OCL statement, coding notes 1-9
OCL statements
/. 1-13
/& 1-12
* 1-13
ASSIGN 1-9
BSCA 1-9
CALL 1-9
CODE MEANING 1-14
COMPILE 1-9
DATE 1-9
FILE 1-9
HALT 1-10
IMAGE 1-10
INCLUDE 1-10
JOB 1-10
LOAD 1-10
LOAD * 1-11
LOG 1-11
NOHALT 1-11
PAUSE 1-11
PRINTER 1-11
PUNCH 1-12
READER 1-12
RUN 1-12
SWITCH 1-12
OMIT parameter for the COPYFILE statement 4-36
operator control commands C-1
operator control commands (OCC) for multiprogramming
cancel 2-54
change 2-54
display 2-54
dump 2-54
halt 2-54
nohalt 2-55
pty 2-55

operator control commands (OCC) for
multiprogramming (continued)
reader 2-55
set 2-55
start 2-56
stop 2-56
operator control commands (see OCC)
options to messages 2-19
organization of the object library 2-28
organization of the source library 2-27
OUTDM statement 4-30
OUTPTX parameter for the COPYFILE
statement 4-34
output file name 4-27
OUTPUT parameter for the COPYFILE
statement 4-34
OUTPUT statement ($HIST) 4-94
overlay linkage editor 2-5, 2-23
overlays
areas 2-7
memory resident 2-7
segments 2-7

PACK parameter
ALT statement 4-8
ALTA statement 4-213
CATLG statement 4-22.2
FILE statement 1-31
REBUILD statement 4-14
REMOVE statement 4-74
VOL statement 4-105
parameter
keyword 1-4
table of parameters 1-14
parameter summary, system service programs
ACCESS statement ($COPY) 4-33
ALLOCATE statement ($MAINT) 4-120
ALT statement ($ALT) 4-8
ALTA statement ($RSALT) 4-213
ASNP statement ($CNFIG) 4-18
AUTHORIZE statement ($QCOPY) 4-174.2
CLASSIFY statement ($QCOPY) 4-174.2
CLEAR statement ($SCOPY) 4-219
COMPRESS statement ($WVTOC) 4-241
COPY statement ($MAINT) 4-132
COPYAREA statement ($SCOPY) 4-177
COPYCTRL statement ($QCOPY) 4-174
COPYFILE statement ($COPY) 4-29
COPYFILES statement ($FCOMP) 4-80
COPYIPL statement ($SCOPY) 4-221
COPYPACK statement ($COPY) 4-29
COPYPACK statement ($DCOPY) 4-63
COPYPCHQ statement ($QCOPY) 4-172
COPYPRTQ statement ($QCOPY) 4-171
COPYQ statement ($QCOPY) 4-174.1
COPYRDRQ statement ($QCOPY) 4-173
COPYSP statement ($QCOPY) 4-171
DELETE statement ($MAINT) 4-146
parameter summary, system service programs
(continued)
DISPLAY statement ($LABEL) 4-110
DISPLAY statement ($QCOPY) 4-174
INSERT statement ($MAINT) 4-152
KEY statement ($COPY) 4-28
MODIFY statement ($MAINT) 4-151
MOVE statement ($SCOPY) 4-220
NAMES statement ($SCOPY) 4-220
NEWNAME statement ($SCOPY) 4-220
OUTDM statement ($COPY) 4-30
OUTPUT statement ($HIST) 4-94
PRINT statement ($HIST) 4-94
REBUILD statement ($BUILD) 4-13
REMOVE statement ($DELETE) 4-72
RENAME statement ($MAINT) 4-154
REPLACE statement ($MAINT) 4-152
RESTORE statement ($QCOPY) 4-174
SELECT statement ($COPY) 4-28
UIN statement ($INIT) 4-101
VOL statement, ($INIT) 4-102
VOL statement, ($TINIT) 4-235
PARITY seven track tape 1-51
partition changing 1-72.1
PARTITION parameter for JOB
statement 1-72.1
partition(s)
multiprogramming 2-57
size 2-8
PAUSE statement
function 1-84
parameters 1-19
PAUSE statement OCL 1-11
permanent file, deleting 1-34
placement of OCL statements (see the
desired statement type)
positional operands C-1
primary initialization 4-100
primary initialization, example 4-106
print compilation date 4-134
print configuration record 4-18.1
PRINT parameter for device independent file
statement 1-64
PRINT statement ($HIST) 4-94
printer chain image (see IMAGE statement)
printer forms (see PRINTER statement)
PRINTER OCL statement
ALIGN parameter 1-19
CLOSE parameter 1-19
COPIES parameter 1-19
DEFER parameter 1-19
DEVICE parameter 1-19
FORMSNO parameter 1-19
LINES parameter 1-19
QCOPY parameter 1-19
PRINTER statement 1-11, 1-85
PRINTER statement parameters 1-19
printing compilation date 4-134
printing file information from the
VTOC 4-109
printing files 4-27
printing files, example 4-44

printing library directories 4-139
printing library directories, example 4-158
printing records using record keys 4-36
printing records using relative record numbers 4-37
printing the entire contents of the VTOC 4-110
priority parameter 1-18
priority parameter, JOB statement 1-72
procedure-name parameter for the CALL statement 1-14, 1-24
procedure override statement 2-33
procedures
  adding a missing parameter 2-33
  adding a statement 2-33
  changing procedure parameters 2-33
  deleting a procedure parameter 2-33
  example 2-34
  inserting statements 2-33
  listing 4-149
  modifying 2-33
  nested 2-35
  normal procedure call 2-32
  procedure override statement 2-33
  removing statements 4-149
  replacing statements 4-149
processing large indexed files 2-23
processing multivolume files 2-24
program function 4-3
program load switch 3-14
program name 4-3
program-name parameter for the LOAD statement 1-18, 1-76
program pack definition 2-39
program pack protection 2-39
program reference information F-1
program(s)
  concepts 2-4
  execution 2-42
  size 2-8
  storing 2-30
programming considerations E-2
programs and file names 1-30, 1-46
programs requiring specific file names 1-30, 1-46
protect program packs 2-39
PTY command C-4
PUNCH statement 1-88
PUNCH statement (OCL) 1-12
PUNCH statement parameters 1-19

QCOPY parameter 1-19
queues
  print 2-67
  punch 2-67
  reader 2-67

read $QCOPY control statements from a file
  considerations and restrictions 4-194
  control statement 4-169
  parameter description 4-181
  parameter summary 4-174
READER command C-4
READER OCC command C-4
READER statement 1-91
READER statement, (OCL) 1-12
READER statement, parameters 1-20
reader-to-library copy function, $MAINT 4-137
reassign a 5444 unit code and change catalog support 4-25
reassign all 5444 unit codes for three partitions 4-25
reassign alternate track program 4-212
reassign one 5444 unit code in a partition 4-25
reassigning simulation areas 1-21, 3-9
REBUILD statement ($BUILD)
  control statement summary 4-12
  DISP parameter 4-14
  LENGTH parameter 4-14
  PACK parameter 4-13
  parameter summary 4-13
  TRACK parameter 4-14
  UNIT parameter 4-13
RECFM parameter, FILE statement (tape) 1-50
RECL parameter, FILE statement (tape) 1-50
RECORDS parameter for the FILE statement (disk) 1-32
records-tracks conversion B-7
recover files ($COPY) 2-15, 4-41
recover index program 2-15, 4-208
REEL parameter
  FILE statement 1-48
  VOL statement 4-235
related publications ii
relationship of OCL to the job stream 1-3
relative location of each 3344 volume 3-10
RELEASE command C-4
RELEASE OCC command C-4
REMOVE statement ($DELET)
  control statement summary 4-72
  DATA parameter 4-74
  DATE parameter 4-74
  LABEL parameter 4-74
  PACK parameter 4-74
  parameter summary 4-73
  UNIT parameter 4-74
REMOVE statement ($MAINT)
  control statement summary 4-150
  functions 4-149
  parameters 4-152
removing files from a disk 4-71
removing gaps from between files 4-78
removing source library statements 4-149
removing temporary library entries 4-146

RENAME statement ($MAINT)
control statement summary 4-153
FROM parameter 4-154
function 4-153
LIBRARY parameter 4-154
NAME parameter 4-154
NEWNAME parameter 4-154
PACKIN parameter 4-154
parameter summary 4-154
renaming a set of source statements in a source library 4-153, 4-165
REORG parameter for the COPYFILE statement 4-36
reorganizing a source library 4-126
reorganizing an object library 4-127
reorganizing the system pack 4-166
REPLACE statement ($MAINT)
control statement summary 4-150
functions 4-149
parameters 4-152
replacing existing library entries 4-138
replacing incorrect data 4-12
replacing source library statements 4-149
replacing statements in a procedure 4-149
replacing the printer chain 1-66
requesting $QCOPY from a terminal 4-205
required tracks for a file index B-4, B-7
required tracks for indexed files B-5, B-7
RESER parameter of MODIFY statement 4-151
reserializing a source library entry 4-149
responding to error messages ($QCOPY) 4-203
RESTART command C-4
RESTART OCC command C-4
restarting checkpointed programs 2-68
restore function 4-79
restore print or punch queue records
considerations and restrictions 4-195
control statement 4-169
parameter description 4-182
parameter summary 4-174
restore print or punch queue records from a file 4-195
restricted names 4-135
restrictions
library maintenance
allocate 4-119
copy 4-135
delete 4-146
modify 4-149
rename 4-153
restrictions concerning assignment of simulation areas 3-9
restrictions concerning assignment of 5444 unit codes 3-9
restrictions on naming library entries 4-135
RETAIN parameter
COPY statement 4-133
DELETE statement 4-148
FILE statement
disk 1-34
tape 1-49
retrieving library entries D-1
REUSE command C-4
REUSE OCC command C-4
RPG II 2-21
rules for coding control statements 4-5
rules for nested procedures 2-37
RUN statement 1-20, 1-92
RUN statement (OCL) 1-12
sample job stream 1-3
sample statements for compiling and storing source programs 2-31
scheduler 2-3
scheduler work area 2-15
scratch file
changing a temporary file to a scratch file 1-34
location 3-10
scratching a file 1-34
SDTF (share define the file) 2-17
SELECT KEY for the SELECT statement 4-36
SELECT PKY for the SELECT statement 4-36
SELECT RECORD for the SELECT statement 4-37
SELECT statement ($COPY)
control statement summary 4-29
FROM parameter 4-36
parameter summary 4-32
SELECT KEY 4-36
SELECT PKY 4-36
SELECT RECORD 4-37
TO parameter 4-36
selecting the 2 option for a message 2-19
selecting the 3 or D option for a message 2-19
SEQFLD parameter of MODIFY statement 4-151
sequence numbers in MODIFY functions 4-149
sequential file tracks B-7
SET command C-5
SET OCC command C-5
setting external indicators 1-93
seven-track tape considerations 1-52
severity codes 2-48
severity codes for NOHALT OCL statement 1-83
SHA printout 4-95
sharing access to added records 2-59

simulation area codes
$SCOPY 4-218
device codes 4-6
simulation area program 4-216
CLEAR parameters
AREA 4-222
CLRNAME 4-181
FROM 4-222
ID 4-222
PACK 4-222
TYPE 4-223
control statement 4-217
COPYAREA parameters
AREA 4-222
FROM 4-222
PACK 4-222
SYSTEM 4-222
TO 4-222
TONAME 4-222
COPYIPL parameters
FROM 4-223
PACK 4-223
TO 4-223
MOVE parameters
AREA 4-224
CLRNAME 4-223
FROM 4-224
PACK 4-224
TO 4-224
TONAME 4-224
NAMES parameter
PRINT 4-224
NEWNAME parameters
AREA 4-223
ID 4-223
newname 4-223
PACK 4-223
TO 4-223
TONAME 4-222
parameter descriptions 4-222
parameter summary
CLEAR 4-219
COPYAREA 4-219
COPYIPL 4-221
MOVE 4-220
NAMES 4-220
NEWNAME 4-220
simulation areas
accessing 3-6
assignment during system generation 3-9
default assignment 3-9
definition of 3-6
maximum number 3-7, 3-10
minimum number 3-7, 3-10
reassignment 1-21, 3-9
restrictions 3-9
use of 3-6
source library
adding entries 4-137
changing size 4-118, 4-124
creating 4-118, 4-124
deleting 4-118, 4-125
source library (continued)
inserting statements 4-149
listing entries 4-149
location 4-124
organization 2-27
physical characteristics
directory 2-27
location 2-27
organization of entries 2-27
size 2-27
removing statements 4-149
reorganizing 4-118, 4-118
replacing statements 4-149
reserializing entries 4-149
SOURCE parameter
ALLOCATE statement 4-121
COMPILE statement 1-14, 1-25
source programs 2-4
special characters 1-5, 4-5
specific file names (disk) 1-30
specific file names (tape) 1-46
spool 2-42, 2-67
spool file copy program
control statement summary 4-168
examples 4-196.6
files and subroutines 4-185
functions 4-168
OCL considerations 4-196.6
parameter descriptions
AUTHORIZE 4-183
CLASSIFY 4-184
COPYCTRL 4-181
COPYPCHQ 4-177
COPYPRTQ 4-176
COPYQ 4-183
COPYRDRQ 4-178
COPYSP 4-176
DISPLAY 4-181
RESTORE 4-182
parameter summary 4-171
program description 4-167
spool file considerations and restrictions
assign class numbers to programs 4-196.3
copy a display of the status of the spool queues 4-194
copy jobs to or from the reader queue 4-192
copy selected job steps from one spool file to another 4-195
copy selected jobsteps from the print queue 4-184.1
copy selected jobsteps from the punch queue 4-190
copy the entire spool file 4-184.1
file requirements 4-184
partition size requirements 4-184
read control statements from a file 4-194
restore print or punch queue records from a file 4-195

spool file copy program (continued)
using the spool file copy program under CCP
CCP assignment set 4-204
considerations for terminating $QCOPY under CCP 4-206
displaying the spool queues 4-203
examples 4-206
how to request $QCOPY from a terminal 4-205
placing jobs on the reader queue from a terminal 4-203
program request 4-202.2
responding to error messages 4-203
user authorization 4-202.3
using $QCOPY from a terminal 4-202.3
SPOOL parameter in JOB statement 1-72
spooling
controlling 2-42
devices used with 2-67
general information 2-67
IPL 2-42
performance with 2-67
processing with 2-43
Standard Character Set A-1
START command C-5
START OCC command C-5
statement
beginning with /,/ 1-6
beginning with other than // 1-7
description, general information 1-8
descriptions (OCL) 1-9
examples (OCL) 1-4
identifier 1-4
length 1-6
STOP command C-5
STOP OCC command C-5
storage medium (disk)
capacity 3-5, 3-6
cylinder designation 3-6
format 3-7, 3-8
storing and compiling source programs in the object library 2-31
storing programs into libraries 2-30
subroutine 15 (SUBR15)
error identification D-6
linking with assembler D-5
linking with RPG II D-2
substitute data 4-13
summary of OCC commands C-1
summary of OCL parameters 1-14
supervisor 2-3
support for cataloging to a program pack 2-39
surface analysis 4-104
SWA-maximum number of files 2-15
switch characters
CALL statement 1-24
INCLUDE statement 1-69
LOAD statement 1-76
SWITCH statement
OCL 1-12
parameter 1-20, 1-93
system control programming (SCP)
date 2-63
history area 2-66
input device 2-64
integrity
data file security 2-69
volume security 2-70
log device 2-65
print device 2-66
punch device 2-66
service programs 2-3
system date 1-27, 2-63
system directory 4-135, 4-145
system history area 2-66
system history area display program ($HIST)
control statement summary 4-94
examples 4-97
function 4-93
OCL considerations 4-94
parameter descriptions 4-94
system input device
changing 1-91
considerations 1-91
system integrity
data file security 2-69
volume recognition facility 2-70
volume security 2-70
system log device 1-79, 2-65
SYSTEM parameter for the ALLOCATE statement 4-121
system print device 1-85, 2-66
system punch device 1-88, 2-66
system service programs 2-3, 4-1
System/360-System/370 packs 4-212

table of OCL statements 1-9
table of parameters 1-14, 1-14
tape error summary program 4-239
tape file statement summary 1-53
tape initialization program 4-233
tape, magnetic
copying using copy/dump program 4-39
error logging 4-239
FILE statement 1-45
FILE statement summary 1-53
initialization 4-233
multifile volumes 1-57
multivolume files 1-55
telling the system not to halt 1-82
telling the system to halt 1-65
temporary file 1-34
temporary file, changing a temporary file to a scratch file 1-34
temporary library entries 4-119
TIME command C-6

TIME OCC command C-5
time stamp messages 4-18.2
TLOG command C-6
TLOG OCC command C-6
TO parameter
ALLOCATE statement ($MAINT) 4-121
COPY statement ($MAINT) 4-133
COPYAREA statement ($SCOPY) 4-222
COPYFILES statement ($FCOMP) 4-80
COPYIPL statement ($SCOPY) 4-223
COPYPACK statement ($COPY) 4-31
COPYPACK statement ($DCOPY) 4-65
COPYQ statement ($QCOPY) 4-183
COPYSP statement ($QCOPY) 4-176
MODIFY function ($MAINT) 4-152
MOVE statement ($SCOPY) 4-224
NEWNAME statement ($SCOPY) 4-223
SELECT statement ($COPY) 4-36
TAPEFILES statement ($FCOMP) 4-81
TRACE command C-5
TRACK parameter for the REBUILD statement 4-16
track requirements, data area B-1
track requirements, index area B-4
track usage for indexed files B-5
TRACKS parameter for the FILE statement 1-15, 1-46
transaction logging E-1
transaction logging (operating considerations) E-2
transaction logging under COBOL and FORTRAN E-1
transaction logging under RPG II E-1
TRANSLATE seven-track tape 1-51
type of information contained in cylinder 0 3-12
TYPE parameter
UIN statement 4-102
VOL statement 4-235
types of directory entries 4-135, 4-135
types of library entries 2-26.1

UIN statement ($INIT)
control statement summary 4-101
ERASE parameter 4-104
parameter summary 4-102
TYPE parameter 4-102
UNIT parameter 4-104
VERIFY parameter 4-104
unit code, meaning 4-18
UNIT parameter
ALT statement ($ALT) 4-8
ALTA statement ($RSALT) 4-213
CALL statement (OCL) 1-24
COMPILE statement (OCL) 1-25
COMPRESS statement ($WVTOC) 4-241
COPYPCHQ statement ($QCOPY) 4-177
COPYPRTQ statement ($QCOPY) 4-176
unit parameter (continued)
COPYRDRQ statement ($QCOPY) 4-178
DISPLAY statement ($LABEL) 4-111
DISPLAY statement ($QCOPY) 4-181
FILE statement (OCL)
device independent 1-64
disk 1-31
tape 1-45
IMAGE statement (OCL) 1-68
LOAD statement (OCL) 1-76
unit1 parameter 1-76
unit2 parameter 1-77
REBUILD statement ($BUILD) 4-13
REMOVE statement ($DELETE) 4-74
UIN statement ($INIT) 4-104
VOL statement ($TINIT) 4-193
unit record restart 2-70
unlabeled tapes 2-25
use of external buffers 2-9
use of main data area 3-10
use of simulation areas 3-6
use of SYSTEM-YES parameter 3-13
using OCL 1-3
using the spool file copy program under CCP 4-202.1

VERIFY parameter
ALT statement 4-8
table of parameters 1-15
UIN statement 4-104
VOL statement ($INIT)
control statement summary 4-101
ID parameter 4-105
NAME360 4-105
OLDPACK parameter 4-105
PACK parameter 4-105
parameter summary 4-102
VOL statement ($TINIT)
control statement summary 4-234
parameters summary 4-235
volume format 3-3
volume recognition facility 2-70
volume, definition of 3-3, 3-4
volumes, maximum number of 2-24
VTOC (volume table of contents)
LABEL parameter 4-111
meaning of information 4-113
printout examples 4-112
VTOC service program
control statement summary 4-241
examples 4-242
function 4-241
OCL considerations 4-242
parameter descriptions 4-241
PACK parameter 4-241
UNIT parameter 4-241
parameter summary 4-241

work area, library maintenance
program 4-122
work files 2-20
work parameter 4-122
writing control statements 4-4
system service programs
coding rules 4-5
END statement 4-5

1403 parameter, LOG statement 1-18, 1-81
1442 parameter
PUNCH statememt 1-88
PUNCH statement 1-19
READER statement 1-20, 1-91
2501 parameter, READER
statement 1-20, 1-91
3340 direct access storage 3-3
3344 direct access storage 3-4
3741 file considerations 4-39

Technical Newsletter

This Newsletter No. GN21-5674
Date 28 September 1979

Base Publication No. GC21-5162-1
File No. S3-36

Previous Newsletters None

IBM System/3 Model 15
System Control Programming
Concepts and Reference Manual

This technical newsletter applies to version 04, modification 00 of the IBM System/3 Model 15 System Control Program (Program Number 5704-SC2) and provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent versions and modifications unless specifically altered. Pages to be inserted and/or removed are:

iii through x
1-9, 1-10
1-21, 1-22
1-29, 1-30
1-55 through 1-58
1-71, 1-72
1-72.1, 1-72.2 (added to accommodate new and moved text)
1-83, 1-84
2-9, 2-10
2-17, 2-18
2-18.1, 2-18.2 (added to accommodate new and moved text)
2-25, 2-26
2-26.1, 2-26.2 (added to accommodate new and moved text)
2-31 through 2-34
2-37, 2-38
2-57, 2-58
2-58.1, 2-58.2 (added to accommodate new and moved text)
2-67, 2-68
2-68.1, 2-68.2 (added to accommodate new and moved text)
4-17, 4-18
4-18.1, 4-18.2 (added to accommodate new and moved text)
4-19, 4-20
4-20.1, 4-20.2 (added to accommodate new and moved text)
4-21, 4-22
4-22.1, 4-22.2 (added to accommodate new and moved text)
4-23, 4-24
4-24.1, 4-24.2 (added to accommodate new and moved text)
4-25, 4-26
4-26.1, 4-26.2 (added to accommodate new and moved text)
4-37 through 4-42
4-65, 4-66
4-77, 4-78
4-83, 4-84
4-93, 4-94
4-131, 4-132
4-141 through 4-144
4-167 through 4-174
4-174.1, 4-174.2 (added to accommodate new and moved text)
4-175 through 4-184
4-184.1, 4-184.2 (added to accommodate new and moved text)
4-185, 4-186
4-186.1, 4-186.2 (added to accommodate new and moved text)
4-187 through 4-194
4-194.1, 4-194.2 (added to accommodate new and moved text)
4-195, 4-196
4-196.1 through 4-196.6 (added to accommodate new and moved text)
4-201, 4-202
4-202.1 through 4-202.4 (added to accommodate new and moved text)
4-203 through 4-206
4-239, 4-240
C-1, C-2
C-2.1, C-2.2 (added to accommodate new and moved text)
C-3, C-4
D-5, D-6
E-1, E-2
F-1, F-2 (added to accommodate new and moved text)
X-1 through X-20

Changes to text and illustrations are indicated by a vertical line at the left of the change.

Summary of Amendments

- Enhancements to Spool File Copy program ($QCOPY)
- Enhancements to Configuration Record program ($CNFIG)
- Additional information about file sharing
- Miscellaneous technical corrections

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901

Printed in U.S.A.