UNISYS

Product Information Announcement

○ New Release ● Revision ○ Update ○ New Mail Code

Title

V Series Systems Operations Guide, Volume 3: System Utilities

This Product Information Announcement announces a revision of the *V Series Systems Operations Guide, Volume 3: System Utilities*. This revision is relative to the 3.3.0 release of the V Series operating system.

This revision includes descriptions of the following new utilities:

- DPKANL, a utility for analyzing and correcting diskpack problems, with the ability to work with diskpack structures at a very low level
- LOADER, a utility for validating print train identifiers and creating custom train printer buffer files
- SQUASH, a utility for consolidating areas on a diskpack
- TAPDIR, a utility for printing lists of the files on magnetic tapes produced by the SYSTEM/COPY, LOADMP, and PACKUP utilities
- TRKTAP, a utility that collects on tape the data produced by the TRAK diagnostic facility
- VFUGEN, a utility that creates electronic vertical forms (EVF) files, which take the place of mechanical vertical formatting units in some buffered printers

The discussion of V Series diskpack subsystems has been updated with a discussion of Version 4 diskpack families.

Various technical and editorial changes have been made to improve the quality and usability of the document.

Changes are indicated by vertical bars in the margins.

To order additional copies of this document

- United States customers, call Unisys Direct at 1-800-448-1424.
- All other customers, contact your Unisys sales office.
- Unisys personnel, use the Electronic Literature Ordering (ELO) system.

Announcement only:

Announcement and attachments:
SD

System: V Series
Release: 3.3.0 May 1995

Part number: 4127 0000–100

UNISYS

V Series Systems

Operations Guide

Volume 3: System Utilities

May 1995

Printed in US America
4127 0000–100

Priced Item

UNISYS

V Series Systems

Operations Guide

Volume 3: System Utilities

Release 3.3.0

May 1995

Printed in US America

Priced Item

4127 0000–100

The names, places, and/or events used in this publication are not intended to correspond to any individual, group, or association existing, living, or otherwise. Any similarity or likeness of the names, places, and/or events with the names of any individual, living or otherwise, or that of any group or association is purely coincidental and unintentional.

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THIS DOCUMENT. Any product or related information described herein is only furnished pursuant and subject to the terms and conditions of a duly executed agreement to purchase or lease equipment or to license software. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such agreement. Unisys cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, special, or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Correspondence regarding this publication should be forwarded to Unisys Corporation either by using the Business Reply Mail form at the back of this document or by addressing remarks to Software Product Information, Unisys Corporation, 25725 Jeronimo Road, Mission Viejo, CA 92691–2792 U.S.A.

V Series Systems

Operations Guide

Volume 3: System Utilities

4127 0000–100

V Series Systems
Operations Guide
Volume 3: System Utilities

4127 0000–100

Bend here, peel upwards and apply to spine.

Page Status

Page	Issue
iii through xi	–100
xii	Blank
xiii through xxvii	–100
xxviii	Blank
xxix through xxxi	–100
xxxii	Blank
xxxiii through xxxiv	–100
1–1 through 1–3	–100
1–4	Blank
2–1 through 2–6	–100
3–1 through 3–37	–100
3–38	Blank
4–1 through 4–7	–100
4–8	Blank
5–1 through 5–30	–100
6–1 through 6–11	–100
6–12	Blank
7–1 through 7–2	–100
8–1 through 8–3	–100
8–4	Blank
9–1	–100
9–2	Blank
10–1 through 10–14	–100
11–1 through 11–3	–100
11–4	Blank
12–1 through 12–6	–100
13–1 through 13–3	–100
13–4	Blank
14–1 through 14–23	–100
14–24	Blank
15–1 through 15–2	–100
16–1 through 16–12	–100
17–1 through 17–11	–100
17–12	Blank
18–1 through 18–2	–100

Unisys uses an 11-digit document numbering system. The suffix of the document number (1234 5678–xyz) indicates the document level. The first digit of the suffix (x) designates a revision level; the second digit (y) designates an update level. For example, the first release of a document has a suffix of –000. A suffix of –130 designates the third update to revision 1. The third digit (z) is used to indicate an errata for a particular level and is not reflected in the page status summary.

Page Status

Page	Issue
19–1 through 19–4	–100
20–1	–100
20–2	Blank
21–1 through 21–10	–100
22–1 through 22–13	–100
22–14	Blank
23–1 through 23–19	–100
23–20	Blank
24–1 through 24–22	–100
25–1 through 25–3	–100
25–4	Blank
26–1 through 26–12	–100
27–1 through 27–4	–100
28–1 through 28–27	–100
28–28	Blank
29–1 through 29–36	–100
30–1 through 30–16	–100
31–1 through 31–34	–100
32–1 through 32–3	–100
32–4	Blank
33–1 through 33–9	–100
33–10	Blank
34–1 through 34–9	–100
34–10	Blank
35–1 through 35–3	–100
35–4	Blank
36–1 through 36–7	–100
36–8	Blank
37–1 through 37–16	–100
38–1 through 38–5	–100
38–6	Blank
Index–1 through–20	–100

About This Guide

Purpose

This guide is written as a companion text to the *V Series Systems Operations Guides, Volumes 1, 2, and 4.* This guide describes the utilities you can use with the Unisys Master Control Program for V Series (MCP/VS) operating system. These utilities perform many functions, including such tasks as copying and moving files, initializing disks and disk packs, and establishing data communications.

Scope

In this guide, the descriptions of each utility or system feature include its purpose, its use, and any commands or messages that are associated with it. In many cases, examples of operator input are provided. In some cases, technical details of program interaction or file layout are included.

Audience

The target audience for this guide is an experienced user of Unisys V Series equipment.

Prerequisites

Anyone using this guide should be familiar with the Master Control Program for V Series (MCP/VS) and the information included in the other operations guides.

Organization

This guide consists of 33 sections that discuss a utility or intrinsic program and 5 sections that discuss system features.

Section 1. B 974LD Intrinsic Program

Use the B 974LD intrinsic program with the B 974 Data Communications Processor.

Section 2. CPLOAD–Communication Processor Firmware Loading Program

The CPLOAD utility program loads firmware to communications processors that are connected to the V Series Communication System (VCS).

Section 3. DISPKV–Disk Pack Utility Program

The DISPKV utility program initializes and purges LAK disk (100-byte-per-sector) and disk pack (180-byte-per- sector) media. The DISPKV program can verify the contents of a disk or disk pack, relocate sectors on a disk or disk pack, change selected fields in a disk pack label, and list the addresses of sectors on a disk or disk pack. Also, the DISPKV program is useful when a media failure occurs.

Section 4. DLPXCO AND DLPXNO–DLP Utility Programs

A data link processor (DLP) is an interface between the V Series processor and a peripheral device. The DLPXCO and DLPXNO utilities dump and clear data link processors that are not functioning because of irrecoverable I/O errors.

Section 5. DMPALL–File Conversion Utility Program

The DMPALL utility program lists, copies, and converts disk, disk pack, and tape files. The DMPALL program can move files from one medium to another. Unlike the SYSTEM/COPY utility program, the DMPALL program can work with files that were not created by a Unisys V Series system, and that have unusual blocking factors, no tape label, or other unusual characteristics. Also, the DMPALL program can list all or part of disk, disk pack, tape, and card files, including object code disk files. In addition, the DMPALL program can produce listings in hexadecimal code, and can produce a translation into characters.

Section 6. DMPANL–MCP Memory Dump Analysis Program

The DMPANL utility program produces a formatted analysis of a memory dump of the MCP.

Section 7. DMPCPY–Memory Dump File Copy Utility Program

The DMPCPY utility program copies a memory dump file from one magnetic tape to another.

Section 8. DMPMEM–Memory Dump Utility Program

The DMPMEM utility program is a floppy-disk based utility that dumps MCP memory to tape. You use the DMPMEM program to produce memory dumps in abnormal situations that cannot be handled by the MCP intrinsic Fault Handler.

Section 9. ECMANL–Environmental Control Module Analysis Utility Program

The ECMANL utility program formats environmental information about the V 500 system and creates a printer backup file.

Section 10. LDCNTL–Pseudo Reader Load Control Utility Program

The LDCNTL utility program creates pseudo card files from other types of files. You can use these pseudo card files to control the system.

Section 11. LDHOST–Host Load Intrinsic Program

The LDHOST utility program downloads firmware (microcode) from the V Series system to devices that are interfaces to peripheral devices. The LDHOST program can download firmware to a programmable controller, a data communication processor (DCP), or a Uniline device.

Section 12. LOADFW–Offline Firmware Loader Program

The LOADFW utility program downloads firmware (microcode) files from the V Series system to certain devices that interface between V Series systems and peripheral devices or between multiple V Series systems. These devices are the disk pack controller, the shared system processor (SSP), and the Uniline data link processor (Uniline DLP).

Section 13. MAKTRN–Translation File Generator Program

The MAKTRN program creates a disk file that is used with the key translation option of the SORT intrinsic utility programs (SORT. and SORT:).

Section 14. MDCOPV–Floppy Disk Copy Utility Program

The MDCOPV utility program copies files to and from floppy disks and performs floppy disk maintenance functions. The MDCOPV program recognizes the industry-compatible minidisk (ICMD) format and the micro-minidisk unit (MMDU) format.

Section 15. MERG:–MCP Merge Intrinsic Program

The MERG: intrinsic program provides file merging functions to utilities written in V Series COBOL ANSI-74.

Section 16. NIFMRG–DCP Utility Program

The NIFMRG utility program merges together files called network information files (NIF). These files contain information used for data communications with the B 874 or B 974 data communications processors, the Offline Reader-Sorter (ORS-DLP), the TELCOM DLP, and CP2000 communications processors. If you plan to use more than one type of these devices at the same time, you must merge the NIF files with the NIFMRG program.

Section 17. PBDPRN–Printer Backup Utility Program

The PBDPRN utility program prints out printer backup files that other utility programs have created.

Section 18. PCOPY–Object Program Copy Utility Program

The PCOPY utility program copies object code files from 100-byte disk to 180-byte disk pack and vice versa.

Section 19. PKCOPY–Disk Pack Copy Utility Program

The PKCOPY utility program copies the data from one disk pack to another.

Section 20. SNPANL–SNAP Analysis Utility Program

The SNPANL utility program takes a raw SNAP file and breaks out the machine state for analysis.

Section 21. SORT.–Sort Intrinsic Program

The SORT. utility program provides rapid sorting facilities for programs written in COBOL ANSI-68 and BPL.

Section 22. SORT:–Tape/Disk Sort Intrinsic Program

The SORT: utility program provides rapid sorting facilities for programs written in COBOL ANSI-74 and for the SRTUTL utility program.

Section 23. SRTUTL–Generalized Sort Utility Program

The SRTUTL utility program is a stand-alone interface for the SORT. utility program. It provides sorting facilities for programs written in RPG and for operators. The SRTUTL program sorts indexed sequential, relative, or sequential files according to keys. Also, the SRTUTL program generates either an ADDROUT or sequential file.

Section 24. SYSTEM/COPY–File Transfer Utility Program

The SYSTEM/COPY utility program copies and converts files from one file storage medium to another, or copies files that are on the same medium. It copies and moves files to and from disk, disk pack, and tape, and also lists the files that reside on various media.

Section 25. SYSUP–Automatic System Recovery Facility

The SYSUP utility program helps to automatically recreate the operating environment when the system fails.

Section 26. UNLODV–Uniline DLP Utility Program

A Uniline DLP is one kind of interface device between a V Series system and a peripheral device. The UNLODV utility program helps prepare a Uniline DLP for use.

Section 27. DCP–Data Communications Processor

You can use a B 874 or B 974 data communications processor (DCP) to reduce the workload on the V Series system involved in data communications. This section contains MCP configuration records, commands, link error codes, and firmware interface error codes for the B 874 and B 974 data communications processors.

Section 28. Debug Facility

The Interactive DEBUG facility enables you to use debug activities (modification of memory, breakpoints, and so forth) for user programs and, optionally, for the operating system.

Section 29. QWIK Disk

QWIK Disk enables you to use part of system memory as RAM disk for rapid-access storage of files.

Section 30. SHARED–Shared Systems and Devices

This section discusses the ability of multiple V Series systems to access the same files on the same media, using a shared system processor (SSP).

Section 31. Pack Subsystems

This section describes the concepts, structure, and operations of pack subsystems for V Series systems.

Section 32. OBJCHK–File Compression Utility Program

This section describes how to use the utility program OBJCHK to compress and decompress a file.

Section 33. VFUGEN–Vertical Format Unit File Builder

This section describes how to use the utility program VFUGEN to create an electronic vertical format (EVF) file that can take the place of a mechanical vertical formatting unit on certain buffered printers.

Section 34. TRKTAP–TRAK to Tape Program

This section describes how to use the utility program TRKTAP to capture diagnostic information produced by the TRAK facility on magnetic tapes.

Section 35. TAPDIR–Tape Directory Report Program

This section describes how to use the utility program TAPDIR to produce reports analyzing the directories of SYSTEM/COPY, LOADMP, and PACKUP tapes.

Section 36. DPKANL–Diskpack Analysis Program

This section describes how to use the utility program DPKANL to diagnose and repair problems with diskpack structures.

Section 37. SQUASH–Diskpack Squash Program

This section describes how to use the utility program SQUASH to consolidate data on a diskpack.

Section 38. LOADER–Train Printer Buffer File Generator

This section describes how to use the utility program LOADER to produce a custom train printer buffer file for use in loading the train printer translate buffer. Also discussed are the standard train printer buffer files.

Results

After reading this guide, you will be able to use those utilities and system features that are relevant to your programming and operating environment.

Related Product Information

This guide is a companion text to the following documents:

V Series Systems Operations Guide Volume 1: Installation (4127 4804)

V Series Systems Operations Guide Volume 2: System Commands (4127 4994)

V Series Systems Operations Guide Volume 4: System Messages and Recovery (4127 0018)

These documents are referred to in the text as Volume 1, Volume 2, and Volume 4, respectively.

The following Unisys documentation may also be of use:

B 2000/B 3000/B 4000/V Series TABSII Installation and Operations Guide (5031735)

B 9290-30 Intelligent Laser Printing System Operator's Guide

B 874/B 974/ORS-DLP MCS Message Headers Programming Reference Manual (5024219)

B 974 Data Communications Processor Software Installation and Operations Guide (5022965)

B 974 Network Definition Language (NDL) Programming Reference Manual (5023104

V Series COBOL ANSI-74 Compiler Programming Reference Manual , Volume 1: Basic Implementation (4127 4945)

V Series COBOL ANSI-74 Compiler Programming Reference Manual , Volume 2: Product Interfaces (4127 4952)

V 300 Maintenance User's Guide

V 500 Reference Manual

V Series CANDE Installation and Operations Reference Manual (4127 0109)
V Series VCS Operations and Utilities Reference Manual (4127 0042)
V Series FLAME Operations Reference Manual (4127 4986)
V Series Systems Software Logging Operations Reference Manual (4127 4812)
V Series TELCOM DLP Installation and Operations Guide (4127 0083)
V Series WFL Compiler Programming Reference Manual (5031461)

Contents

Section 15. MERG:—MCP Merge Intrinsic Program

Section 16. NIFMRG—DCP Utility Program

Section 17. PBDPRN—Printer Backup Utility Program

Section 18. PCOPY—Object Program Copy Utility Program

Section 19. PKCOPY—Disk Pack Copy Utility Program

Section 20. SNPANL—SNAP Analysis Utility Program

Section 21. SORT.—Sort Intrinsic Program

Section 22. SORT:—Tape/Disk Sort Intrinsic Program

Section 23. SRTUTL—Generalized Sort Utility Program

Section 24. SYSTEM/COPY—File Transfer Utility Program

Section 29. QWIK Disk

Section 30. SHARED—Shared Systems and Devices

Section 31. Pack Subsystems

Section 32. OBJCHK—File Compression Utility Program

Section 33. VFUGEN—Vertical Format Unit File Builder

Section 34. TRKTAP—TRAK to Tape Program

Section 35TAPDIR—Tape Directory Report Program

Figures

Figures

Tables

Tables

Section 1
B 974LD Intrinsic Program

Overview

The B 974LD program on the host communicates with the HOST LOAD program on the B 974 data communications processor (DCP). The program transfers files, executes Network Definition Language (NDL) compiles, and initializes and activates the communications network. This section gives an overview of the B 974LD program features.

The *B 974 DCP Software Installation and Operations Guide* provides procedures for initiating the B 974LD program.

The host initiates the B 974LD program in the following situations:

- When you set the MCP DCP option with the SO command and the DCP option or when the system sets the MCP DCP option because the system configuration file contains a USE record with the DCP option
- When you use the LH command to reactivate the B 974
- When the host finishes precompiling a Network Definition Language (NDL) source file and the compilation process needs to be completed on the B 974
- When a message from the B 974 must be displayed on the host operator display terminal (ODT)

The B 974LD program communicates with only one B 974 at a time. Therefore, hosts with multiple B 974s can have multiple copies of the B 974LD program running simultaneously. The program performs the following functions:

- Transfers various files, including printer backup listings, B 974 firmware code files, B 974 system memory dumps, and B 974 resident program dumps to the host
- Transfers the B 974 system maintenance logs (MLOGs) and printer backup files generated by B 974MCS trace and other utilities to the host
- Transfers B 974 NDL source code files, firmware code files, and host message control system (MCS) information files from the host to the B 974
- Initiates and monitors NDL compiles on the B 974 from the host
- Displays information from the B 974 on the host ODT
- Activates the data communications network

Network Initialization

When you use the LH, LH (with the WARM option), or SO (with the DCP option) commands to execute the B 974LD program, or when there is a USE record (with the DCP option) in the system configuration file, the MCP attempts to initialize and activate the data communications network.

- If you use the LH command, the B 974LD program transfers the firmware code file from the host to the B 974.
- If you use the LH (with the WARM option) command or the SO (with the DCP option) command, the B 974LD program transfers the firmware code file from the host to the B 974 only if the firmware code file on the host is not the same as the firmware code file on the B 974. If the firmware code file is the same on both the host and the B 974, the B 974LD program does not transfer the firmware code file.

Recovery of Messages

Messages are lost in the following situations:

- If the host fails (all messages queued on the host are lost).
- If a firmware code file is transferred to the B 974 when you initialize it (all messages queued on that DCP are lost). If the file is not transferred, the messages are retained. To retain the messages, use either the LH (with the WARM option) command or the SO (with the DCP option) command. (Note that you will lose the messages if the system transfers a firmware file, regardless of the command you use.) To discard the messages, use the LH command.

For more information on when the B 974LD program transfers the firmware code file, refer to the *B 974 DCP Software Installation and Operations Guide.*

Whenever you execute a message control system (MCS) that was previously active, all messages queued in the B 974 during the host failure are sent to the MCS.

Additional Tasks

After network initialization or NDL compilation, the B 974LD program performs the tasks described in the following paragraphs.

System and Program Dump Files

The B 974LD program uploads to the host system memory dumps or B 974 resident program dumps.

B 974 System Maintenance Logs

The B 974LD program analyzes maintenance log (MLOG) files generated by the B 974 and uploads the listings to the host.

Printer Backup Files

The B 974LD program uploads printer backup files from the B 974 disk to the host. The files could have been generated by an MCS issuing a trace function header.

NDL Compilation

The B 974 NDL compilation process has two stages, one on the host and one on the B 974. When the stage on the host is completed with no errors, the B 974LD program transfers an intermediate code file to the B 974. The B 974 then completes the compilation, which results in a firmware code file.

If the compilation on the B 974 contains no errors, the B 974LD program transfers a copy of this completed firmware code file to the host for safekeeping. For more information about this process, refer to the *B 974 DCP Software Installation and Operations Guide.*

Section 2
CPLOAD—Communication Processor Firmware Loading Program

Overview

CPLOAD is a utility program provided as part of the V Series Communication System (VCS). CPLOAD loads firmware and files to communication processors (CPs) that are connected to the host system through VCS. CPLOAD can load the following communication processors:

- **TELCOM DLP.** CPLOAD provides the same functions as the LDHOST utility provides in a non-VCS environment.
- **CP 368X Front End Processor.** CPLOAD provides the functions that the CP 368X-specific programs FDPLOD and FDPRES provide in a non-VCS environment. These functions include CP 368X firmware, system, and form file downloading.
- **B 874 Data Communications Processor.** CPLOAD provides the same functions as the LDHOST utility provides in a non-VCS environment.
- **B 974 Data Communications Processor.** CPLOAD provides the same functions as the B 974LD utility provides in a non-VCS environment.

Initiating CPLOAD

The CPLOAD program begins under the following conditions:

- The MCP detects a label containing the value FE*IBL in the input received from a CP 368X.
- VCS software requests the operating system to initiate the CPLOAD program.
- You enter an LH command, using the CP <CP number> or the NCP <NCP number> syntax option.

  Note: *Use the LH command only for debugging. Unisys does not recommend its use for normal operations.*

LH Command

You can initiate the CPLOAD program with VCS-dependent options of the LH command. The CPLOAD program performs specific actions in response to other syntax options of the LH command.

WARM Option

When you enter the LH command with the WARM option, the CPLOAD program compares the versions of the firmware files residing on the CP and on the host system. The CPLOAD program loads only those firmware files that are present in *different* versions on the CP and the host system.

When you do not enter the WARM option, all firmware files are unconditionally loaded to the CP.

DUMP Option

The CPLOAD program responds differently to the DUMP option of the LH command, depending on the type of CP:

- For TELCOM DLPs and B 874 DCPs, the CPLOAD program dumps CP memory into a file on the host system *before* it loads the firmware.
- For CP 368Xs, the CPLOAD program transfers the system parameter file of the CP 368X into a file on the host system before it loads the firmware.
- For B 974 DCPs, no special action is taken.

In any case, the program names the host system file CxxxyP, where xxx represents the last three digits of the CP number and y represents the system number of the host system on which the CPLOAD program is executing. For example, if you enter the LH command LH CP 14 DUMP on system number 2, the CPLOAD program creates a dump file named C0142P.

You can analyze dump files from TELCOM DLPs or B 874 DCPs with the DC1ANL utility program. You can analyze system parameter files from CP 368Xs with the VCSPRN utility program. Refer to the *V Series Communication System (VCS) Operations Reference Manual* for more information.

FE*IBL Label (CP 368X only)

When the MCP detects a label with the value FE*IBL in the input from a CP 368X, it performs the functions of the CP 368X-specific program FDPLOD. The CPLOAD program loads only a code file into the memory of the CP. This process is called *host booting the CP*.

VCS Initiation

VCS software can respond to a wide variety of conditions in a data communications network. In several situations, VCS instructs the operating system to execute the CPLOAD program by passing specific instructions that it can interpret. The VCS software can request that certain specific files be unconditionally loaded to the CP. When the CPLOAD program receives such a request, it determines if any additional files must be loaded to the CP and then loads all the required files.

Error Messages

The CPLOAD program or a network controller (NC) can display any of the following messages:

`CP IS NOT IN A LOADABLE STATE`

The CPLOAD program cannot load files to the CP. This error message usually appears together with the following message:

```
ERROR FOR FEP REQUEST: FC = 14, SFT = 05, EC = 02, P1 = 0000
```

Depending on runtime conditions, you could choose to boot the CP or ignore the error. However, if the system file timestamps mismatch you must boot the CP.

`CP 368X FORMS LIST (FEPRES) NOT FOUND`

This message means that the forms list used for loading the CP 368X resident forms must be application number 3.

`SYSTEM FILE UPLOAD ABORTED: FILE XXXXXX ON XXXXXX`

An unrecoverable error has occurred during a dump of a CP 368X. Any information transferred to the host system before the dump was aborted is retained.

`FEP REQUEST ERROR: FC = XX, SFT = XX, EC = XX, P1 = XX`

This message appears when the CP 368X detects an error during a function request. The function code, the sub-function, the error code, and parameter 1 fields and their values also appear on the ODT.

`INAPPROPRIATE REQUEST FOR TELCOM`

This message comes from a TELCOM DLP in response to a load request.

`INCORRECT CP: SOUGHT # XXXXXX, FOUND # XXXXXX`

Certain data in the VCS initialization file is incorrect. This message appears when the network controller (NC) fails during an update, before the update is complete. The SOUGHT field shows the number of the CP that the CPLOAD program was looking for; the FOUND field shows the number of the CP that the CPLOAD program was able to find.

`INCORRECT INDEX: SOUGHT # XXXXXX, FOUND # XXXXXX`

Certain data in the VCS initialization file is incorrect. This message appears when the network controller (NC) has failed during an update, before the update was complete. The SOUGHT field shows the number of the CP that the CPLOAD program was looking for; the FOUND field shows the number of the CP that the CPLOAD program was able to find.

INIT FILE CONTAINS NO RECORDS: XXXXXX ON XXXXXX

The CPLOAD program is unable to read any information from the VCS initialization file. The CPLOAD program could be attempting to read a non-VCS file. This message appears infrequently and indicates that the operating system passed incorrect information about the location of the VCS initialization file to the CPLOAD program.

INIT FILE NOT FOUND: XXXXXX ON XXXXXX

The VCS initialization file is not present. This message appears infrequently and indicates that the operating system passed incorrect information about the location of the VCS initialization file to the CPLOAD program.

INIT FILE: SOUGHT XXXXXXXXXXXX, FOUND XXXXXXXXXXXX

Certain data in the VCS initialization file is incorrect. This message appears when the network controller (NC) fails during an update, before the update is complete. The SOUGHT field shows the type of VCS entity that the CPLOAD program was searching for; the FOUND field shows the type of entity that the CPLOAD program was able to find.

INSUFFICIENT TELCOM MEMORY FOR REQUESTED FUNCTION

This message comes from a TELCOM DLP in response to a load request.

INVALID FORMAT: XXXXX BY XXX: XXXXXX ON XXXXXX

The VCS initialization file is in the wrong format. This message appears infrequently and indicates that the operating system passed incorrect information about the location of the VCS initialization file to the CPLOAD program.

INVALID INIT FILE KEY: XXXXXXXX, TYPE XXXXXXXXXXX

Certain data in the VCS initialization file is incorrect. This message appears when the network controller (NC) fails during an update, before the update is complete. The KEY field shows the key value the CPLOAD program was searching for; the TYPE field shows the type of VCS initialization record that the CPLOAD program was looking for.

LINE PROCESS NOT AVAILABLE FOR PROTOCOL # XXXXXX

A TELCOM line references a protocol for which there is no line process. The possible causes are the following:

- TELCOM does not support this protocol.
- The wrong line process file was specified in the CP record of the initialization file.

NET FILE: INVALID HEADER: ERROR DIGIT INDEX = XXXX

This message indicates a specific Network File error that can occur. The error digit index gives the position of the incorrect digit in the Network File direct I/O header. Possible causes for this error include invalid data passed to the CPLOAD program by the operating system, invalid data in the VCS initialization file, or inconsistencies within the CPLOAD program.

NET FILE: <error text>

This error relates to the Network File and the CPLOAD program displays it.

NO HOST CONNECTED TO THIS CP CLAIMS TO BE MASTER

The CPLOAD program cannot find a master host system in the VCS initialization file for the specified CP. This message appears infrequently.

DUMP ABORTED: FILE XXXXXX ON XXXXXX

An unrecoverable error has occurred during a dump of a TELCOM DLP. Any information transferred to the host system before the dump was aborted is retained.

TELCOM FUNCTION NOT DONE, NO REASON GIVEN

This message comes from a TELCOM DLP in response to a load request. The TELCOM has not returned any data to the CPLOAD program.

TELCOM LINE OR STATION MUST BE NOT READY

This message comes from a TELCOM DLP in response to a load request.

TELCOM TABLE NOT DELETED, OUTPUT QUEUE IS NOT EMPTY

This message comes from a TELCOM DLP in response to a load request.

TIMESTAMP OF ZERO FOUND IN HOST FILES

A timestamp with a value of zero exists in one or more of the host system VCS files. This message appears infrequently. Possible causes for this error include an incorrect file name in the CP or HOST records of the VCS initialization file, or a problem with the network controller (NC).

UNRECOGNIZED INITIALIZATION PARAMETERS VERSION

A mismatch has occurred between the versions of CPLOAD and the operating system. The mismatch is caused by differing release levels of the operating system and the CPLOAD program.

```
UNRECOGNIZED RBI RECEIVED FROM TELCOM, RBI = xx
```

This message comes from a TELCOM DLP in response to a load request.

```
XXXXXXXXXXXX: EXPECTED TO FIND XXXXXX, FOUND XXXXXX
```

Certain data in the VCS initialization file is incorrect. This message appears when the network controller (NC) has failed during an update, before the update was complete.

The first field in the message shows the VCS entity type (for example STATION), the EXPECTED TO FIND field shows the expected number of entities, and the ACTUALLY FOUND field shows the actual count. This message indicates an error condition, but the CPLOAD program continues loading firmware files.

```
ZERO OR NON-MATCHING TIMESTAMPS RETURNED
```

The timestamps of the firmware files on the CP do not match the timestamps of the firmware files on the host, even after the CPLOAD program has loaded new firmware files.

This message appears infrequently. Possible causes for this error include a CP in a state where file loading is invalid, or errors detected by the CP during the loading of files. When files are loaded to a CP 368X, the CPLOAD program does not detect any errors, but the operating system does display error messages.

Section 3
DISPKV—Disk Pack Utility Program

Overview

The DISPKV utility program performs disk and disk pack initialization functions. The specific functions performed during each execution are determined by the command and by other supporting data input such as a string either through the ODT or on cards.

DISPKV is used to format certain disks or disk packs. The Initialization, Verification, and Relocation (IVR) functions are used to format a disk or disk pack. For a list of the disk packs for which DISPKV is used, refer to Table 31–1 in Section 31, "Pack Subsystems."

Use DISPKV only for version 1 disk packs. For version 2 and greater disk packs, use the ALTER NEW PACK system command. For a complete description of the various disk pack versions, refer to Section 31, "Pack Subsystems." The ALTER NEW PACK command is discussed in the *V Series Systems Operations Guide Volume 2: System Commands.*

The major functions that you can perform with the DISPKV program include the following:

- Initializing virgin media as version 1 disk pack (180-byte sectors)
- Initializing virgin media as disk (100-byte sectors)
- Reinitializing previously initialized media (100- or 180-byte sectors) as disk pack (180-byte sectors)
- Reinitializing previously initialized media (100- or 180-byte sectors) as disk (100-byte sectors)
- Reinitializing a selected cylinder or a range of cylinders of a previously initialized disk or version 1 disk pack
- Relocating a selected sector on a disk or version 1 disk pack
- Verifying all sectors on a disk or version 1 disk pack
- Labeling a version 1 disk pack
- Changing selected fields in a version 1 disk pack label
- Purging a version 1 disk pack
- Listing the addresses of all relocated sectors on a disk or disk pack

Execution Syntax

Figure 3–1 shows the two forms of the syntax you use to execute the DISPKV program.

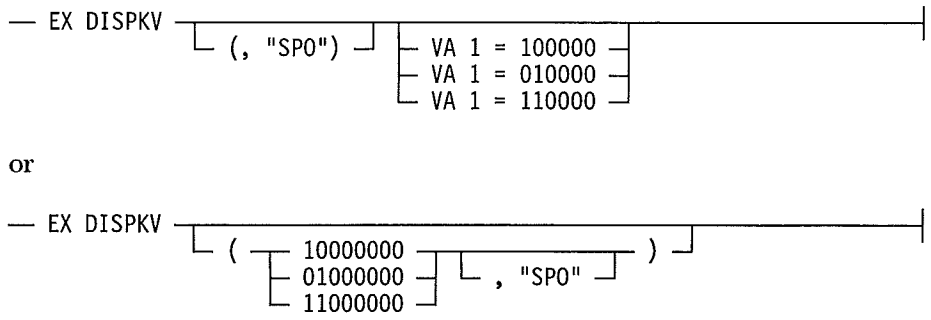


Figure 3–1. DISPKV Syntax

EX DISPKV

Use this option to execute the DISPKV program. When you execute the DISPKV program, you must provide parameters to indicate what you want it to do. The default input medium for these parameters is a card file called INPUTF.

(,"SPO")

Use this option to specify that you will provide the parameters for the DISPKV program with the AX command on the ODT.

If the program encounters an error during ODT input, a detailed set of instructions appears on the terminal screen to explain how to correct the input error.

VA 1=100000

You must use this value statement, which sets switch 1, when you perform the following functions:

- Request a full IVR on a 235, 207, or 659 disk pack.
- Convert a 235 or 207 disk pack from 100-byte to 180-byte format.
- Convert a 235 or 207 disk pack from 180-byte to 100-byte format.
- Convert a 207 or 659 disk pack from sequential to interlaced mode.
- Convert a 207 or 659 disk pack from interlaced to sequential mode.

This value statement forces a short (one pass) IVR operation for all disk pack types except 207, 235, 659, 680, 682, MD4, and MD8.

VA 1=010000

Use this value statement, which sets switch 2, to eliminate retries during verification. When switch 2 is set, the system relocates any errors found during verification and does not perform any retries.

VA 1=110000

Use this value statement to set both switches 1 and 2. Refer to VA 1=100000 and VA 1=010000 earlier in this section for more information about setting switches.

Figure 3–2 shows the syntax for the DISPKV program parameters that you enter on the ODT.

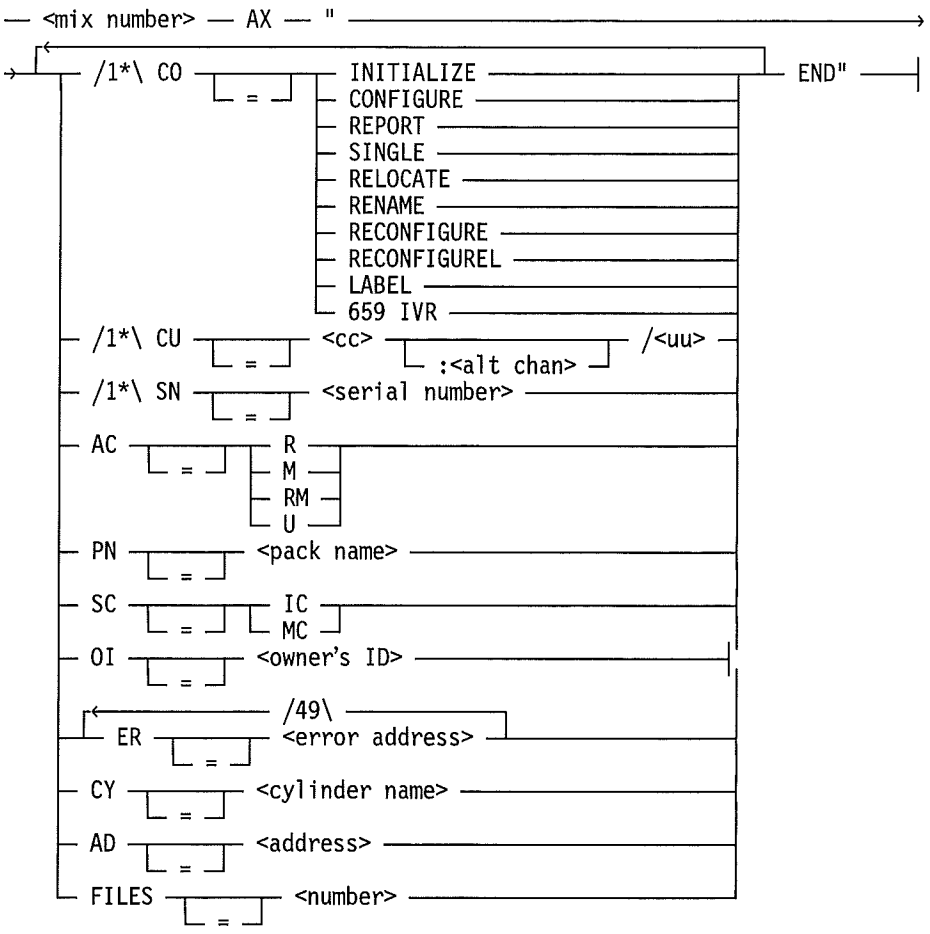


Figure 3–2. DISPVK General Syntax

<mix no> AX "

After you execute the DISPKV program with the (,"SPO") option, you enter parameters, using the AX command on the ODT. You must enclose the entire parameter string in quotation marks. All data input within the quotation marks is freeform.

Precede each parameter with a two-letter keyword that identifies the parameter. An equal sign is optional between the keyword and the parameter. If you do not place an equal sign after the keyword, you must follow it with at least one blank.

Table 3–1 shows the DISPKV program keywords, their meanings, and their default values (if any). If you want to use the default value, do not enter the keyword associated with that value.

Table 3–1. DISPKV Parameter Keywords and Default Values

Keyword	Meaning	Default
CO	Command	None
CU	Channel/unit	None
SN	Serial number	None
AC	System access code	U (Unrestricted)
PN	Pack name	SYS180
SC	System interchange code	MS (Medium Systems)
OI	Owner's-id	All blanks
ER	Error address	None
CY	Cylinder number	None
AD	Address	None
FILES	Files	110

CO

This keyword identifies the command parameter. All subsequent keywords and parameters are related to the command parameters until the next CO, and you can enter them in any order. Each command is described in its own railroad diagram under "Commands" later in this section. Table 3–4 in this section identifies the functions that each DISPKV program command performs.

Examples

```
CO=INITIALIZE ...

CO RECONFIGURE ...
```

CU <cc>:<alt chan>/<uu>

This keyword identifies the parameter for the channel, alternate channel, and unit numbers.

All three parts of the parameter must be numeric. Channel and unit are required, but the alternate channel portion of the parameter is optional. You can declare only one alternate channel in the CU parameter. The alternate-channel declaration, if specified, must be preceded by a colon.

The alternate-channel declaration enables the DISPKV program to use an alternate I/O path to a disk pack subsystem. To declare an alternate channel to a disk pack, refer to the DISK, DLP, and EXCHANGE records in Volume 1. After you execute the DISPKV program, you can use the CU parameter to indicate that the disk pack can be accessed with the alternate channel. You can use the alternate channel to reduce competition for the primary channel, when you perform lengthy DISPKV commands such as INITIALIZE and CONFIGURE.

The channel and alternate channel must be valid disk pack DLPs, and the unit must be a unit number 0–15. If you do not specify an alternate channel, the DISPKV program uses the primary channel. The channel, alternate channel, and unit that you have specified must be declared on the system and must have a hardware type of DPK or DSK.

Examples

```
... CU=8/2 ...
```

If the MCP configuration file includes DLP 7 DSK, DLP 8 DSK, or EXCHANGE 8 7, then the following could be the DISPKV input:

```
... CU 8:7/2 ...
```

SN <serial number>

This keyword identifies the serial-number parameter. The serial number is a 1- to 6-digit numeric combination. It cannot be all zeros.

Examples

```
... SN=12345 ...

... SN 123456 ...
```

AC RM
AC R
AC M
AC U

The AC keyword identifies the parameter for the system access code. This optional parameter specifies the disk pack type and access restriction.

The available access codes and their names are the following:

- RM (Restricted Master). Family names are required to access files on this type of disk pack subsystem. If you do not assign another name to this disk pack with the PN parameter, you must use the default family name SYS180 to access the files on this string of disk packs.
- R (Restricted). This code assigns continuation disk packs to a restricted master disk pack. The family names assigned to these continuation disk packs must match the name of the string for the restricted master disk pack.
- M (Master). This code declares master disk pack for unrestricted system resource use. Family names are not required to access files on this type of disk pack subsystem.

- U (Unrestricted). This value is the default, and it declares continuation disk packs to an unrestricted disk pack subsystem. Family names are not required to access files on this disk pack.

Disk packs with system access codes of M or RM are base disk packs. Disk packs with system access codes of R or U are continuation disk packs. Restricted disk packs (system access code of R or RM) can contain only files having a multifile ID identical to the disk pack name. The syntax is <family name>/<file ID>. For example, if a disk pack has a system access code of RM and a disk pack name (family name) of FAM, all files written to that disk pack must be specified as FAM/<file ID>.

You can assign files in either of the following formats as unrestricted disk packs (system access code of U or M): <pack name>/<file ID> or <file ID>.

If you do not include the keyword AC in the string, the program uses the default access code of unrestricted. If you set the system interchange code to IC, then the system access code must be unrestricted.

Examples

```
... AC=R ...

... AC RM ...
```

PN <pack name>

This keyword identifies the disk pack name parameter. The disk pack family name can be up to 17 characters long, but the MCP recognizes only first six characters. The name cannot be all blanks, nor can it contain special characters or embedded blanks. The default disk pack name is SYS180.

The disk pack name is also called the family name for that disk pack. Files located on disk pack are referred to by a multifile ID consisting of <family name>/<file name>. For example, a file named BAND on a disk pack named ZEPHER would be referenced as ZEPHER/BAND.

Multiple disk packs can have the same disk pack names but must have different serial numbers. The system designates one as the base disk pack and the others as continuation disk packs. The master disk pack has a system access code of RM (restricted master) or M (master). Continuation disk packs have a system access code of R (restricted) or U (unrestricted).

A disk pack name of DISK, PACK, or DSKn (where n is any single character, including a space) is not allowed. These names are reserved for the MCP.

Examples

```
... PN=JAKE ...

... PN BACKUP ...
```

SC IC
SC MS

This keyword identifies the parameter for the system interchange code. This optional parameter specifies how you intend to use the disk pack. When it is set to interchange (IC), you can use the disk pack to interchange files between various families of Unisys

systems. When it is set to medium system (MS), you can use the disk pack on Unisys V Series systems. The default value for SC is MS.

Example

```
... SC=MS ...
```

OI <owner's ID>

This keyword identifies the owner's ID parameter. The owner's ID is a field in the disk pack label. It can be up to 14 characters long with no embedded blanks. The default value for the owner's ID is all blanks.

Examples

```
... OI=DOCGROUP ...

... OI ZEPHYRBAND ...
```

ER <error address>

This keyword identifies the error address parameter. Use this optional parameter to identify, by address, possible bad spots on the disk media. The DISPKV program relocates these addresses to spare sectors.

Each 225-type disk pack has a paper label attached. The label lists addresses that were found to be marginal by factory IVR functions.

When you use the DISPKV INITIALIZE command, enter the listed addresses in the form ER=CCCC/HH/SSS:

CCCC = The cylinder number of the error address

HH = The head number of the error address

SSS = The sector number of the error address

You can enter a maximum of 50 addresses.

Examples

```
... ER=24/7/65 ER 228/8/6 ...

... ER 12/3/31 ...
```

Note: *Any bad areas on 235-type disk packs have already been relocated and MUST NOT be input to DISPKV.*

CY <cylinder number>

This keyword identifies the cylinder number parameter. This parameter is required for the SINGLE command and is optional for the 659IVR command. The parameter specifies the cylinder or cylinders to be reinitialized. The cylinder number can be 1 to 4 digits in length and range between zero and the maximum number of cylinders on the disk pack. Table 3–2 shows the maximum number of cylinders on each type and mode of disk pack.

Table 3–2. Maximum Number of Cylinders

Type	Mode	Cylinders
225	In 100- or 180-byte mode	405
235	In 100- or 180-byte mode	811
206	Interlaced in 100- or 180-byte mode	814
206	Sequential in 100- or 180-byte mode	814
206	Binary Interlaced in 100- or 180-byte mode	814
206	Binary Sequential in 100- or 180-byte mode	814
207	Decimal Interlaced in 100- or 180-byte mode	1397
207	Decimal Sequential in 100- or 180-byte mode	1397
207	Binary Sequential in 100- or 180-byte mode	1563
207	Binary Interlaced in 100- or 180-byte mode	1563
659	Binary Sequential in 100- or 180-byte mode	2241
659	Binary Interlaced in 100- or 180-byte mode	2241
677	Binary Sequential in 100- or 180-byte mode	822
677	Binary Interlaced in 100- or 180-byte mode	822
680 †	Binary Sequential in 180-byte mode	884
682 †	Binary Sequential in 180-byte mode	1768
MD4 †	Storage Module Device (SMD) 180-byte mode	821
MD4 †	Storage Module Device (SMD) 100-byte mode	821
MD8 †	Extended Storage Module Device (XSMD) 180-byte mode	823
M9710 †	SCSI device in 180-byte mode	1494

† The SINGLE command is not allowed with these drive types.

Examples

```
... CY=108 ...

... CY 228 ...
```

AD <address>

This keyword identifies the address parameter. This parameter is required for the RELOCATE command. It specifies the decimal sector address to be relocated. The address can be 1 to 6 digits in length and range between zero and the highest addressable sector on the disk pack. Table 3–3 shows the highest addressable sector on each type of disk pack.

Table 3–3. Highest Addressable Sector

Type	Mode	Address
225	In 100-byte mode	654974
225	In 180-byte mode	485169
235	In 100-byte mode	969144
235	In 180-byte mode	969144
206	Decimal Interlaced in 100-byte mode	565729
206	Decimal Sequential in 100-byte mode	476189
206	Decimal Interlaced in 180-byte mode	362229
206	Decimal Sequential in 180-byte mode	321529
206	Binary Interlaced in 100-byte mode	565729
206	Binary Sequential in 100-byte mode	476189
206	Binary Interlaced in 180-byte mode	362229
206	Binary Sequential in 180-byte mode	321529
207	Decimal Interlaced in 100-byte mode	998854
207	Decimal Sequential in 100-byte mode	998854
207	Decimal Interlaced in 180-byte mode	998854
207	Decimal Sequential in 180-byte mode	998854
207	Binary Interlaced in 100-byte mode	1117544 †
207	Binary Sequential in 100-byte mode	1117544 †
207	Binary Interlaced in 180-byte mode	1117544
207	Binary Sequential in 180-byte mode	1117544
659	Binary Interlaced in 100-byte mode	4696989 †
659	Binary Sequential in 100-byte mode	3957129 †
659	Binary Interlaced in 180-byte mode	3015489
659	Binary Sequential in 180-byte mode	2679189
677	Binary Interlaced in 100-byte mode	2185064 †
677	Binary Sequential in 100-byte mode	1841050 †
677	Binary Interlaced in 180-byte mode	1403214
677	Binary Sequential in 180-byte mode	1246844

continued

Table 3–3. Highest Addressable Sector (cont.)

Type	Mode	Address
680 ‡	Binary Sequential in 180-byte mode	2413320
682 ‡	Binary Sequential in 180-byte mode	4829369
MD4 ‡	Storage Module Device (SMD) in 180-byte mode	682259
MD4 ‡	Storage Module Device (SMD) in 100-byte mode	682259
MD8 ‡	Extended Storage Module Device (XSMD) in 180-byte mode	1389179
MD8 ‡	Extended Storage Module Device (XSMD) in 100-byte mode	1389179 †
M9710 ‡	SCSI device in 180-byte mode	3746952

† The MCP can address up to only 999999 sectors in the 100-byte mode on these devices.

‡ The RELOCATE command is not allowed with these drive types.

Examples

```
... AD=8105 ...

... AD 680738 ...
```

FILES <number>

This keyword identifies the optional files parameter. You can use FILES only with the LABEL command. It specifies the number of files expected on the disk pack.

When you use the files parameter, the DISPKV program constructs the appropriate number of contiguous directory sector entries for the specified number of files. Because the entire directory will be on a contiguous area of disk pack, head movement is minimized during directory use.

If you do not use the files parameter, the DISPKV program assigns a default directory size that is large enough for 110 files. If the directory fills up, the MCP expands it by assigning additional areas, as available, elsewhere on the disk pack.

You must enter the entire word (FILES).

Examples

```
... FILES 3758 ...

... FILES 14284 ...
```

END

This option terminates ODT input. You can continue to enter AX commands until you enter *END*. The keyword END and the final quotation mark signal the end of the input data for this ACCEPT statement.

Examples

```
12AX "END"
```

Commands

The following pages describe and give the railroad diagrams for each of the DISPKV program commands. A command is identified by the CO keyword.

INITIALIZE Command

The INITIALIZE command is the most comprehensive DISPKV command. It purges and completely recreates the entire disk pack. INITIALIZE performs the following three major functions:

- **Initialization.** This function directs the disk pack controller to write sector addresses and gaps in all sectors on the disk pack. Then the program writes a predefined data pattern in the data field of each sector.
- **Verification.** This function directs the controller to read and check all the addressable sectors on the disk pack for address errors and protection-code errors. In addition, the program compares the data field of each sector against the predefined data pattern to check for validity.
- **Relocation.** This function specifies that if an error is detected during verification, it is retried 10 times. If the error is still present after the tenth retry, the program relocates the sector in error to a spare sector. The relocate procedure writes the address of the sector error into a spare sector address field and writes the predefined data pattern into the data area of the spare sector.

Together these functions are abbreviated IVR. After each relocation, the system repeats the Verification function to check for additional errors. The system permits a maximum of five relocates per cylinder. If errors occur after the fifth relocate, the system removes the rest of the cylinder from the master available table (XPed), starting with the error address just received.

The system permits a maximum of 2000 XPs for each disk pack. After 2000 XPs, a message appears on the ODT stating the total number of errors and the cylinder number of the last error. You can then terminate the DISPKV program or continue it. If you terminate the DISPKV program, the total number of errors received for each read/write head appears on the ODT. If you continue the DISPKV program, the program continues to remove sectors from the master available table.

When the IVR function is complete, the DISPKV program creates the following items:

- A disk pack label that contains the serial number, system access code, pack name, system interchange code, and owner's ID
- A master available table that contains all usable space on the disk pack
- An available table that contains all space not removed (XPed)
- An empty disk pack directory table

In addition, the DISPKV program generates an error summary report if it encounters any errors during the IVR process. This report contains any error result descriptors obtained and the I/O descriptor fired to the device.

Because of the error checking and the size of the disk packs, the INITIALIZE command can take from 60 to 300 minutes. Processing time varies, depending on the job mix, the channel activity, and the type of disk pack being initialized.

Figure 3–3 shows the syntax for the DISPKV INITIALIZE command.

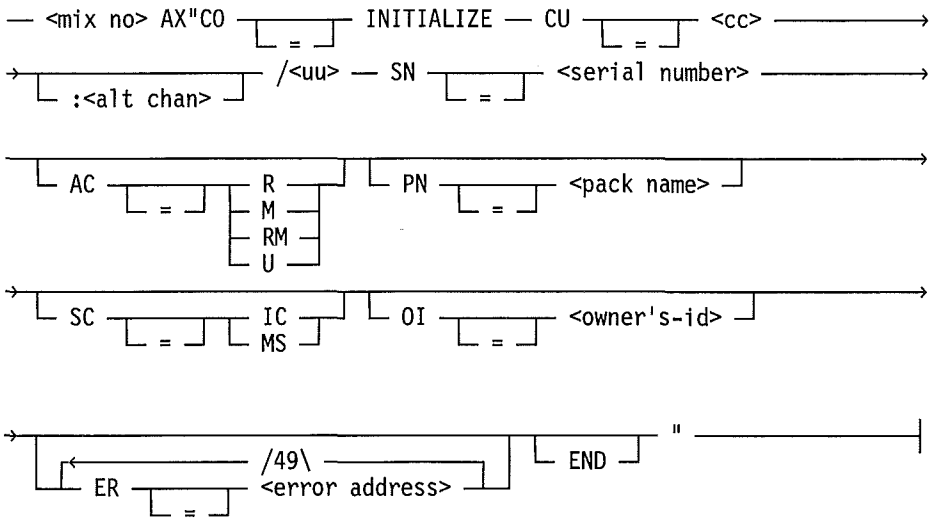


Figure 3–3. DISPKV INITIALIZE Syntax

Required Parameters

The DISPKV program requires the following parameters:

- Command
- Channel/unit
- Serial number

The program uses default values for all parameters that it does not find in the input string. Detailed information regarding these parameters and their defaults appears under "Execution Syntax" earlier in this section.

Restrictions

The INITIALIZE command is invalid for 680, 682, MD4, and MD8 disk packs. To perform the IVR functions on a 680 or 682 disk pack, use the peripheral test driver (PTD) program named M68IVR. To perform the IVR functions on MD4 or MD8 disk pack, use the PTD program named SMDIVR. Then use the LABEL command of the DISPKV program to create the disk pack label, the master available table, the available table, and the empty directory. PTD programs are available from your Unisys Customer Service Engineering representative.

Use the INITIALIZE command when you are converting 100-byte disk packs to 180-byte disk packs and 180-byte disk packs to 100-byte disk packs. You also use INITIALIZE when you are converting interlaced to sequential mode or vice versa. When you are converting 207, 235, or 659 disk packs, refer to the value statements that are described under "Execution Syntax" in this section.

INITIALIZE may or may not be valid for 659 disk packs, depending on the situation:

- If you used the DISPKV program to perform the IVR functions on a 659 disk pack (relocating a maximum of five sectors on each cylinder), INITIALIZE is valid.
- If you used the PTD program named SEQIVR (also called the *shared spares* version of PTD IVR) to relocate more than five sectors, INITIALIZE is invalid.

Addresses are not written on 235 or 207 disk pack, but data fields are initialized.

The disk pack directory and available tables are created only on 180-byte disk packs.

Example

```
8AX "CO INITIALIZE CU 8/2 SN 7142 AC=RM PN=MYPACK OI=MARIBEL"

8AX "CO INITIALIZE CU 6/4 SN 28391 END"
```

In this example, the syntax causes the DISPKV program to perform the IVR functions on two disk packs. The first one has a serial number of 7142 and is on unit 2 connected to DLP 8. It will be initialized as a restricted master with a disk pack name of MYPACK and an owner's ID of MARIBEL. The other parameters will assume default settings.

The second disk pack has a serial number of 28391 and is on unit 4 connected to DLP 6. All optional parameters will assume default values.

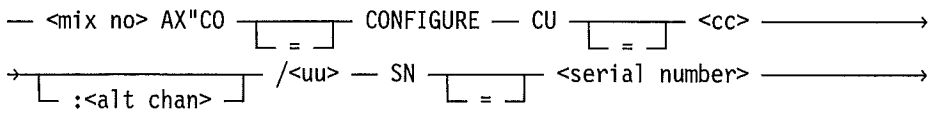
At the conclusion of the second INITIALIZE, the DISPKV program will go to end-of-job.

CONFIGURE Command

The CONFIGURE command purges information and performs Verification and Relocation functions on disk packs that are either preinitialized from the factory or have been previously initialized with the DISPKV INITIALIZE command. The preinitialized disk packs include the 207, 235, and 659 types.

The CONFIGURE command checks all addressable sectors on the disk pack for errors. The program relocates any sectors in error to spare sectors. Any previously relocated sectors remain relocated. If the disk pack being checked is in 180-byte mode, the program creates the disk pack label, available table, and directory table. If errors occur, the program generates an error-summary report.

Figure 3–4 shows the syntax for the DISPKV CONFIGURE command.



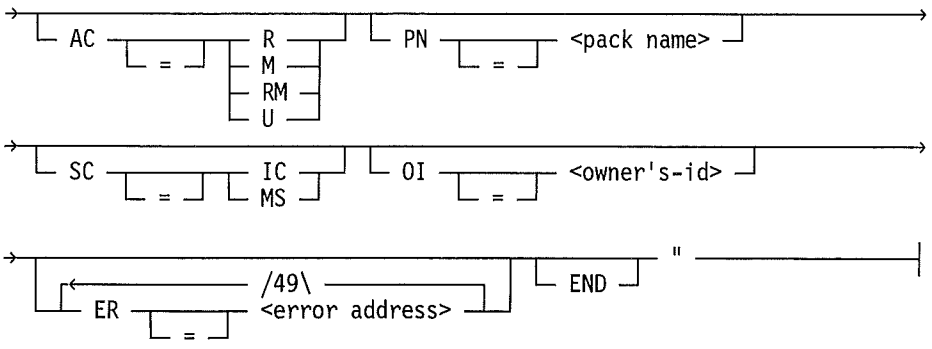


Figure 3–4. DISPKV CONFIGURE Syntax

Required Parameters

The DISPKV program requires the following parameters:

- Command
- Channel/unit
- Serial number

The program uses default values for all parameters that it does not find in the input string. Detailed information regarding these parameters and their defaults appears under "Execution Syntax" earlier in this section.

Restrictions

The disk pack tables are created only on 180-byte disk packs.

The CONFIGURE command is invalid for 680, 682, MD4, and MD8 disk packs. To perform the IVR functions on a 680 or 682 disk pack, use the peripheral test driver (PTD) program named M68IVR. To perform the IVR functions on MD4 or MD8 disk packs, use the PTD program named SMDIVR. PTD programs are available from your Unisys Customer Service Engineering representative.

The CONFIGURE command may or may not be valid for 659 disk packs, depending on the situation:

- If you used the DISPKV program to perform the IVR functions on a 659 disk pack (relocating a maximum of five sectors on each cylinder), CONFIGURE is valid.
- If you used the PTD program named SEQIVR (also called the *shared spares* version of PTD IVR) to relocate more than five sectors, CONFIGURE is invalid.

REPORT Command

The REPORT command scans all spare sectors on the disk pack and produces a printed report of all relocated sectors.

Figure 3–5 shows the syntax for the DISPKV REPORT command.

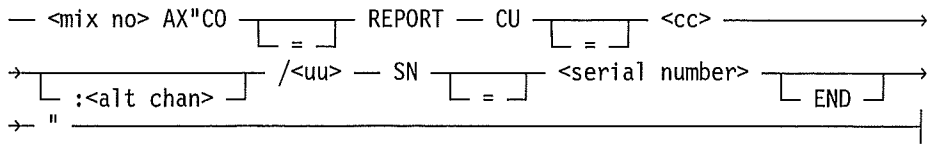


Figure 3–5. DISPKV REPORT Syntax

Required Parameters

The DISPKV program requires the following parameters:

- Command
- Channel/unit
- Serial number

All other parameters in the input string are checked for valid syntax, but are not used. Detailed information regarding these parameters and their defaults appears under "Execution Syntax" earlier in this section.

Restrictions

The REPORT command is invalid for 680, 682, MD4, and MD8 disk packs.

The REPORT command may or may not be valid for 659 disk packs, depending on the situation:

- If you used the DISPKV program to perform the IVR functions on a 659 disk pack (relocating a maximum of five sectors on each cylinder), REPORT is valid.
- If you used the peripheral test driver (PTD) program named SEQIVR (also called the *shared spares* version of PTD IVR) to relocate more than five sectors, REPORT is invalid.

SINGLE Command

The SINGLE command performs IVR functions on a single cylinder of a 100-byte or 180-byte disk pack. The disk pack label, tables, and other data on the disk pack are not affected.

You can use SINGLE if a particular cylinder on an initialized pack has an excessive number of errors or if the addresses are corrupted and you do not want to perform a full disk pack IVR. The program writes all addresses, error protection codes, and predefined data patterns on each sector of the specified cylinder. Then the program reads the sectors to check for address errors, protection code errors, and data validity. The program relocates any bad sectors to spare sectors in the cylinder.

Before initializing and verifying a cylinder, the DISPKV program reads all the cylinder spares. If any of the spare sectors are in use, an ODT display identifies the relocated sectors and asks whether they are to be relocated after the cylinder is reinitialized. When the DISPKV program completes the IVR functions, you can relocate any previously

relocated sectors again. The DISPKV program also relocates any ER addresses supplied during the process.

When the DISPKV program completes the entire SINGLE process, an ODT display identifies all relocated sectors and other sectors that you need to remove.

Figure 3–6 shows the syntax for the DISPKV SINGLE command.

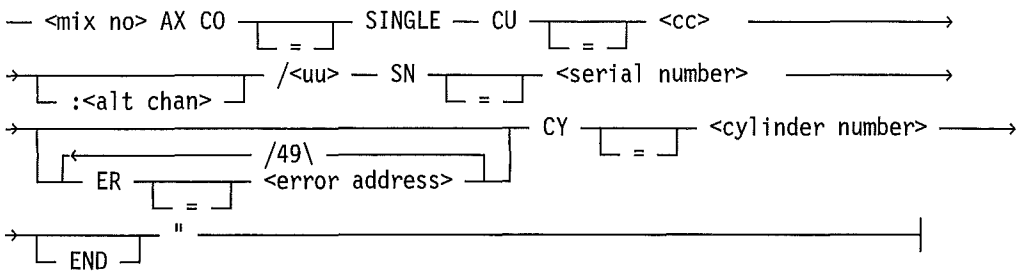


Figure 3–6. DISPKV SINGLE Syntax

Required Parameters

The DISPKV program requires the following parameters:

- Command
- Channel/unit
- Serial number
- Cylinder

In addition, you can use the address parameter with the SINGLE command to identify addresses to be relocated. All other parameters in the input string are checked for valid syntax but are not used. Detailed information regarding these parameters and their defaults appears under "Execution Syntax" earlier in this section.

Restrictions

To perform a single cylinder IVR, you must reserve the entire cylinder to be processed with the SINGLE command, using the XP system command.

The SINGLE command is invalid for 680, 682, MD4, and MD8 disk packs. To perform a single cylinder IVR on a 680 or 682 disk pack, use the peripheral test driver (PTD) program named M68IVR. To perform a single cylinder IVR on MD4 or MD8 disk packs, use the PTD program named SMDIVR. PTD programs are available from your Unisys Customer Service Engineering representative.

The SINGLE command may or may not be valid for 659 disk packs, depending on the situation:

- If you used the DISPKV program to perform the IVR functions on a 659 disk pack (relocating a maximum of five sectors on each cylinder), SINGLE is valid.

- If you used the PTD program named SEQIVR (also called the *shared spares* version of PTD IVR) to relocate more than five sectors, SINGLE is invalid.

RELOCATE Command

The RELOCATE command takes the specified decimal address and relocates it into a spare sector on the cylinder if one is available. You can use this procedure when a sector on the disk pack produces an excessive number of errors and when you do not want a single cylinder or full disk pack IVR.

If no sector is available, the program aborts the request and a message appears on the ODT. After the DISPKV program relocates the sector, the program performs a verify pass against the entire cylinder to check for errors in the relocation process. If the relocated sector has errors, the program retries the relocation five times. After the fifth unsuccessful attempt, a message appears on the ODT and the system terminates the process.

The disk pack label, tables, and all other data on the disk pack are not affected by the RELOCATE command.

Figure 3–7 shows the syntax for the DISPKV RELOCATE command.

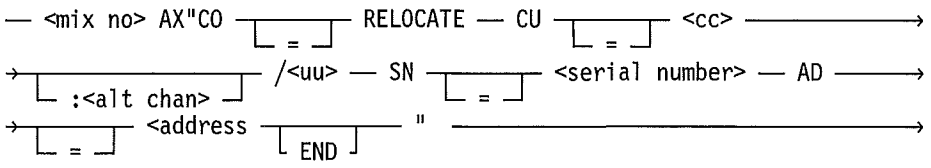


Figure 3–7. DISPKV RELOCATE Syntax

Required Parameters

The DISPKV program requires the following parameters:

- Command
- Channel/unit
- Serial number
- Address

All other parameters in the input string are checked for valid syntax, but are not used. Detailed information regarding these parameters and their defaults appears under "Execution Syntax" earlier in this section.

Restrictions

The address to be relocated must be reserved using the XP system command.

The RELOCATE command is invalid for 680, 682, MD4, and MD8 disk packs. To relocate an address on 680 or 682 disk packs, use the Peripheral Test Driver (PTD) program named M68IVR. To relocate an address on MD4 or MD8 disk packs, use the PTD program named

SMDIVR. PTD programs are available from your Unisys Customer Service Engineering representative.

RELOCATE may or may not be valid for 659 disk packs, depending on the situation.

- If you used the DISPKV program to perform the IVR functions on a 659 disk pack (relocating a maximum of five sectors on each cylinder), RELOCATE is valid.
- If you used the PTD program named SEQIVR (also called the *shared spares* version of PTD IVR) to relocate more than five sectors, RELOCATE is invalid.

RENAME Command

The RENAME command is valid only for a 180-byte disk pack. The RENAME command changes various fields on the disk pack label without purging the disk pack or changing any tables. You can change the following with the RENAME command:

- Pack name
- System access code
- Owner's ID

Figure 3–8 shows the syntax for the DISPKV RENAME command.

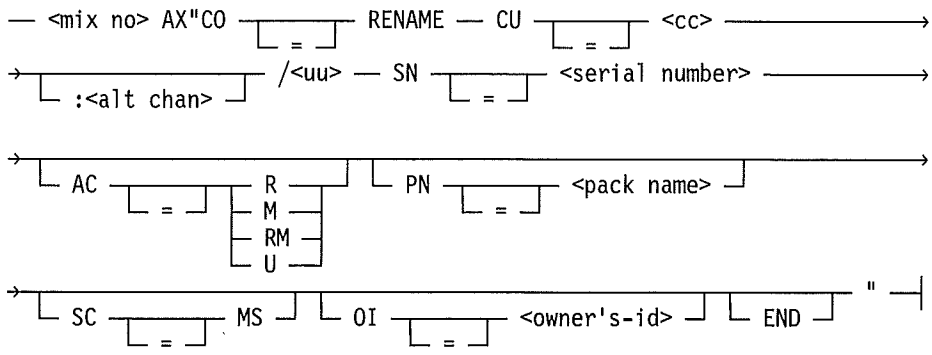


Figure 3–8. DISPKV RENAME Syntax

Required Parameters

The DISPKV program requires the following parameters:

- Command
- Channel/unit
- Serial number

If disk pack name, system access code, and owner's ID appear in the input string, then the program updates the label to these values. If they are not included in the input string, the program assigns default values to these parameters.

All other parameters in the input string are checked for valid syntax, but are not used. Detailed information regarding these parameters and their defaults appears under "Execution Syntax" earlier in this section.

Restrictions

The serial number must match the existing serial number in the label. The system interchange code, if included, must equal MS. Otherwise, the RENAME command aborts.

RECONFIGURE Command

The RECONFIGURE command is valid only for 180-byte disk pack. The RECONFIGURE command purges all files on the disk pack and creates a new available table and empty directory, using the existing master available table. The disk pack label is not changed.

Figure 3–9 shows the syntax for the DISPKV RECONFIGURE command.

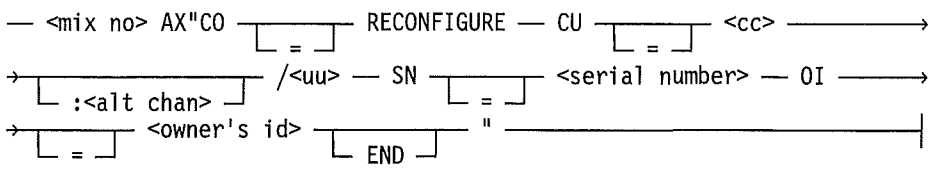


Figure 3–9. DISPKV RECONFIGURE Syntax

Required Parameters

The DISPKV program requires the following parameters:

- Command
- Channel/unit
- Serial number
- Owner's ID

  Note: *Owner's ID is required only if you specified it when you executed the DISPKV program.*

All other parameters in the input string are checked for valid syntax, but are not used. Detailed information regarding these parameters and their defaults appears under "Execution Syntax" earlier in this section.

Restrictions

If the serial number and owner's ID parameters do not match the existing serial number and owner's ID in the disk pack label, the RECONFIGURE command aborts.

RECONFIGUREL Command

The RECONFIGUREL command is valid only for 180-byte disk pack. The RECONFIGUREL command purges all files on the disk pack and creates a new available table and empty directory, using the existing master available table. You can change the disk pack name, system access code, and owner's ID fields in the disk pack label.

Figure 3–10 shows the syntax for the DISPKV RECONFIGUREL command.

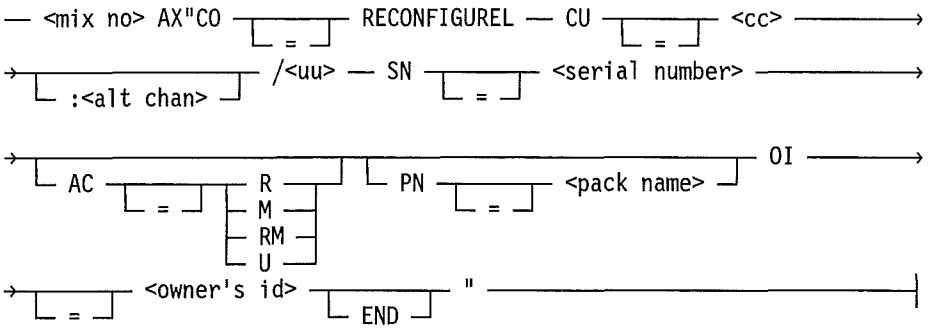


Figure 3–10. DISPKV RECONFIGUREL Syntax

Required Parameters

The DISPKV program requires the following parameters:

- Command
- Channel/unit
- Serial number
- Owner's ID

  Note: *Owner's ID is required only if you specified it when you executed the DISPKV program.*

If disk pack name and system access code appear in the input string, the program updates the label to these values. If they are not included in the input string, the program assigns default values to these parameters. Detailed information regarding these parameters and their defaults appears under "Execution Syntax" earlier in this section.

LABEL Command

The LABEL command is valid only for 180-byte disk pack. The LABEL command purges all files on the disk pack and creates a new label, master available table, available table, and empty directory. You can change the system access code, disk pack name, system interchange code, and owner's ID fields in the disk pack label.

Figure 3–11 shows the syntax for the DISPKV LABEL command.

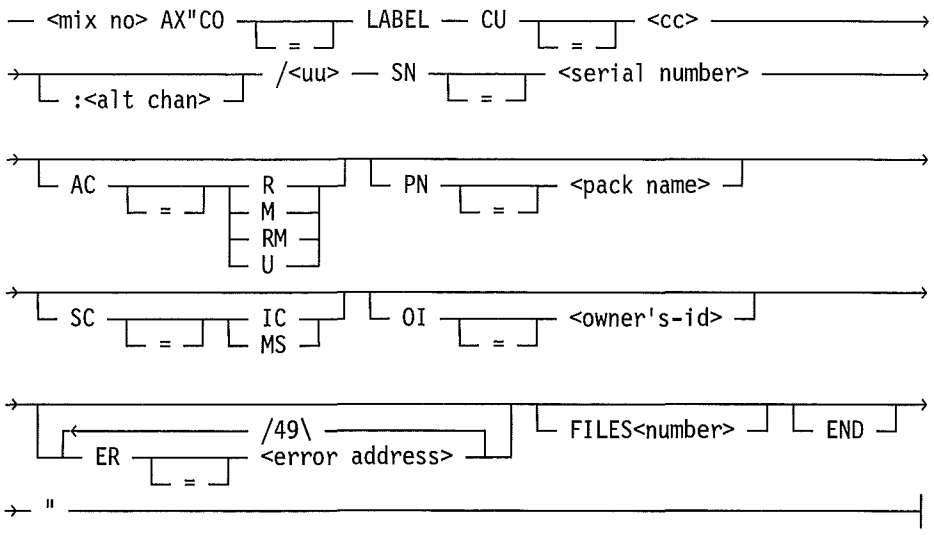


Figure 3–11. DISPKV LABEL Syntax

Required Parameters

The DISPKV program requires the following parameters:

- Command
- Channel/unit
- Serial number

If the system access code, disk pack name, system interchange code, or owner's ID appear in the input string, then the program creates a new label, using these values. The program assigns default values to any of these parameters that are not included in the input string.

You can use the files parameter to establish a contiguous directory table large enough to hold the maximum number of files expected to be put on the disk pack. Detailed information regarding these parameters and their defaults appears under "Execution Syntax" earlier in this section.

Restrictions

The LABEL command does not verify the disk pack and it does not perform error correction. The program will ignore all previously unavailable (XPed) sectors when it builds the new tables.

659IVR Command

The 659IVR command performs an IVR on a single cylinder, on a selected range of cylinders, or on all cylinders of any type of disk pack except 680, 682, MD4, or MD8. The command unconditionally retains all previously relocated sectors and relocates any new sectors found in error.

Depending on the range of cylinders chosen, the 659IVR command may or may not purge all files on the disk pack and create a new label, master available table, available table, and empty directory. You can change the system access code, disk pack name, system interchange code, and owner's ID fields in the disk pack label.

Figure 3–12 shows the syntax for the DISPKV 659IVR command.

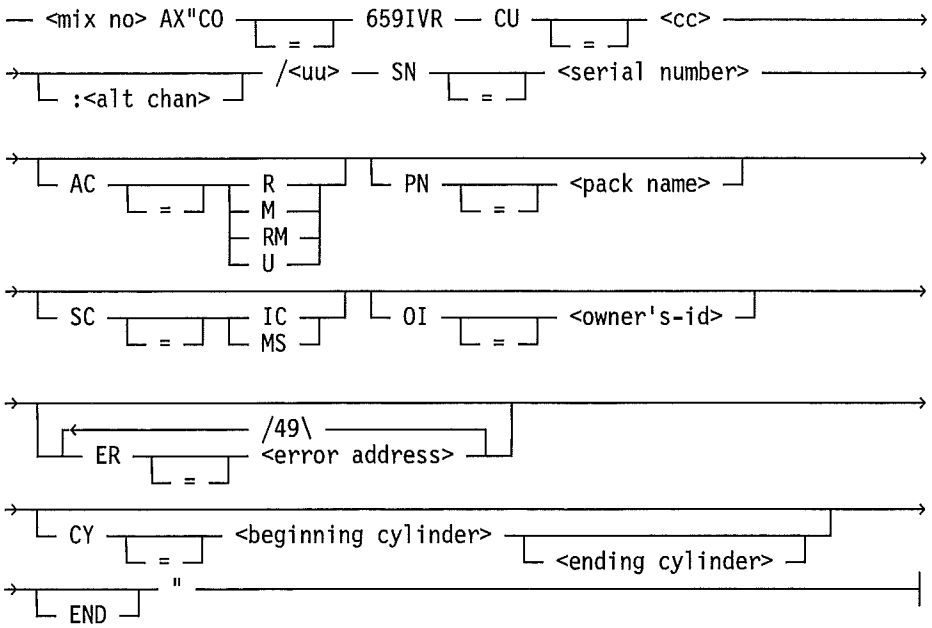


Figure 3–12. DISPKV 659IVR Syntax

Required Parameters

The DISPKV program requires the following parameters:

- Command
- Channel/unit
- Serial number

If cylinder 0, which contains the label information, is to be initialized, then the system access code, disk pack name, system interchange code, and owner's ID should appear in the input string to be included in the new label. The program assigns default values to any of these parameters not included in the input string. Detailed information regarding these parameters and their defaults appears under "Execution Syntax" in this section.

The value of the beginning cylinder is the number of the cylinder to be initialized. The value of the ending cylinder is the number of the last cylinder to be initialized. This parameter is valid only if you have specified the beginning cylinder parameter.

- If you do not specify the beginning parameter, the program initializes the entire disk pack.

- If you use the ending cylinder, separate it from the beginning cylinder with a dash.
- If you omit the ending cylinder parameter or if you specify a value that is greater than the number of cylinders on the disk pack, then the program performs the initialization through the last cylinder.
- If you do not specify any cylinder numbers, then the DISPKV program asks you to specify whether initialization is to begin with cylinder 0. If your response to this request is yes (Y), then the DISPKV program performs the IVR functions on all cylinders on the disk pack. Otherwise, it performs an IVR of the first through the last cylinders on the disk pack.
- If you specify a range of cylinder numbers beginning with cylinder 0, then the DISPKV program asks you to confirm that you want to include cylinder 0. If your answer is no (N), the program follows the original request for all cylinders except cylinder 0.

Restrictions

The 659IVR command is invalid for 680, 682, MD4, and MD8 disk packs. You must use peripheral test driver (PTD) programs to initialize them.

Table 3–4 shows the DISPKV command functions.

Table 3–4. DISPKV Command Functions

	Init	Cnfig	Rpt	Sgl	Reloc	Rename	Recnf	Recnfl	Label	659IVR
Writes all sector addresses	Y	N	N	N	N	N	N	N	N	Y/N
Checks all addressable sectors	Y	Y	N	N	N	N	N	N	N	Y/N
Relocates sectors in error	Y	Y	N	N	N	N	N	N	N	Y
Purges all files	Y	Y	N	N	N	N	Y	Y	Y	Y/N
Creates new disk pack label	Y	Y	N	N	N	N	N	N	Y	Y/N
Updates existing disk pack label	N	N	N	N	N	Y	N	Y	N	N
Creates new master available table	Y	Y	N	N	N	N	N	N	Y	Y/N
Creates new available table	Y	Y	N	N	N	N	Y	Y	Y	Y/N
Creates new disk pack directory table	Y	Y	N	N	N	N	Y	Y	Y	Y/N
Creates error summary report if errors are found	Y	Y	N	N	N	N	N	N	N	Y
Creates report of all relocated sectors	N	N	Y	N	N	N	N	N	N	N
Writes sector addresses in a single cylinder	N	N	N	Y	N	N	N	N	N	Y/N
Checks addressable sectors in a single cylinder	N	N	N	Y	N	N	N	N	N	Y/N
Relocates sectors in error in a single cylinder	N	N	N	Y	N	N	N	N	N	Y/N
Relocates specified sectors	N	N	N	N	Y	N	N	N	N	N

Responding to ODT Input Messages

If the DISPKV program asks for a specific input, for example, serial number or channel/unit, then you should enter only the data it requests. You must not reenter the keyword. Do not enter the keyword END, if the program requests specific input. If you enter END in this case, the program ignores it.

Displaying Status Information

During execution, you can set a switch that causes the DISPKV program to display status information. Use the SW system command to set switch 8 to one of three values.

You can use SW8 to interrogate the DISPKV program for either the cylinder number of the disk pack that is being initialized or configured, the number of errors that have been detected, or both. Table 3–5 shows the syntax and results.

Table 3–5. SW8 Uses

<mix no>SW8=1	Displays the current cylinder number
<mix no>SW8=2	Displays the total number of errors encountered on a basis of each read/write head
<mix no>SW8=3	Displays both the current cylinder number and the number of errors

Initializing M9710, MD4, MD8, 680, and 682 Disk Pack Types

You cannot use the DISPKV program to initialize M9710, MD4 (SMD), MD8 (XSMD), 680, or 682 disk pack types. You must use the PTDMNO program that is on the MCP system tape.

Before attempting to initialize media with the PTDMNO program, load and print the PTDMNI, M68FVI, SMDIVI, SCSIVI, and SCSIVR files to disk from the MCP system tape. These files contain documentation on the operation of the PTDMNO program and are in printer backup disk format.

You can use the DISPKV program to rename and relabel MD4, MD8, 680, and 682 disk pack.

Disk Pack and Controller Styles Supported

The DISPKV program supports many different models of disk pack controllers and disk packs.

The program supports the following controllers:

- BX383 DPDC through an HT-DLP
- BX384 DPDC through an HT-DLP
- BX385 DPDC through an HT-DLP
- B 9387 DPDC through an HT-DLP, or SEQ-HT-DLP
- B 9389 DPDC through an SEQ-HT-DLP

Note: *SMD and XSMD disks do not use a disk pack controller. They are connected directly to a DLP.*

Tables 3–6 and 3–7 show the disk and disk pack styles that the DISPKV program supports.

Table 3–6. Available Disk (100-Byte) Styles

Style	Type	Notes
B 9486–4	225	Dual spindle, removable
B 9484–8	235	Dual spindle, removable
B 9484–5	206	Dual spindle, removable
B 9494–41	207	Dual spindle, fixed
B 9484–12	677	Memorex removable
B 9494–5	659	Memorex fixed
MD4–2 or –4	MD4	SMD fixed

Table 3–7. Available Disk Pack (180-Byte) Styles

Style	Type	Notes
B 9486–4	225	Dual Spindle, removable
B 9484–8	235	Dual spindle, removable
B 9484–5	206	Dual spindle, removable
B 9494–41	207	Dual spindle, fixed
B 9484–12	677	Memorex removable
B 9494–5	659	Memorex fixed
B 9494–12	680	Memorex fixed, dual
B 9494–24	682	Memorex fixed, dual actuator–double density
MD–2 or –4	MD4	SMD fixed
MD8–2 or –4	D8	XSMD fixed–double density
M9710	M9710	SCSI fixed

Tables 3–8 and 3–9 show the DISPKV commands that are valid for each disk and disk pack type.

Table 3–8. DISPKV Commands and Qualified Disk (100-Byte) Types

	Init	Cnfig	Rpt	Sgl	Reloc	Rename	Recnf	Recnfl	Label	659IVR
225	Yes	Yes	Yes	Yes	Yes	N/A	N/A	N/A	N/A	Yes
235	Yes	Yes	Yes	Yes	Yes	N/A	N/A	N/A	N/A	Yes
206	Yes	Yes	Yes	Yes	Yes	N/A	N/A	N/A	N/A	Yes
207	Yes	Yes	Yes	Yes	Yes	N/A	N/A	N/A	N/A	Yes
677	Yes	Yes	Yes	Yes	Yes	N/A	N/A	N/A	N/A	Yes
659	Yes	Yes	Yes	Yes	Yes	N/A	N/A	N/A	N/A	Yes
MD4	No	No	No	No	No	N/A	N/A	N/A	N/A	No

Table 3–9. DISPKV Commands and Qualified Disk Pack (180–Byte) Types

	Init	Cnfig	Rpt	Sgl	Reloc	Rename	Recnf	Recnfl	Label	659IVR
225	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
235	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
206	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
207	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
677	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
659	Yes †	Yes †	Yes †	Yes †	Yes †	Yes	Yes	Yes	Yes	Yes
680	No	No	No	No	No	Yes	Yes	Yes	Yes	No
682	No	No	No	No	No	Yes	Yes	Yes	Yes	No
MD4	No	No	No	No	No	Yes	Yes	Yes	Yes	No
MD8	No	No	No	No	No	Yes	Yes	Yes	Yes	No
M9710	No	No	No	No	No	Yes	Yes	Yes	Yes	No

† These five functions of the DISPKV program may or may not be valid for 659 disk packs, depending on the situation. They are valid if you have used the DISPKV program to perform the IVR functions on the 659 disk pack (relocating a maximum of five sectors on each cylinder). They are invalid if you have used the peripheral test driver (PTD) program named SEQIVR (also called the *shared spares* version of PTD IVR) to relocate more than five sectors, these functions of DISPKV are invalid.

File Equate Information

When you execute the DISPKV program, you may need to know the following internal and external names used by the program for all its input and output files:

- DISPKV input card file
  - Internal name INPUTF
  - External name INPUTF
- DISPKV output error summary report
  - Internal name PRINTF
  - External name PRINTF
- DISPKV output DLP error report
  - Internal name DLPERR
  - External name DLPERR

Input Examples

The DISPKV program permits both ODT and card input. The following examples show valid ODT input. Figure 3–13 shows valid card input.

Valid ODT inputs are the following:

```
EX DISPKV (,"SPO")
```

Enter input for the DISPKV program. You must enclose the entire line with quotation marks. You must enter END to stop ACCEPT commands.

```
**DISPKV=002 ACCEPT

2 AX "CO CONFIGURE CU 8/1 SN 2 CO INITIALIZE"

**DISPKV=002 ACCEPT

2 AX "CU 8:7/5 SN 11 AC RM ER 1/1/1 ER 2/3/4"

**DISPKV=002 ACCEPT

2 AX "PN JAKES OI IVRSUN"

**DISPKV=002 ACCEPT

2 AX "CO SINGLE CU 8/4 SN 875 CY 300 ER 300/6/29"

**DISPKV=002 ACCEPT

2 AX "CO RELOCATE CU 8/6 SN 39 AD 385472"

**DISPKV=002 ACCEPT

2 AX "CO CONFIGURE CU 6:5/4 END"
```

Results of Input

If each of the units that you have designated is a 180-byte disk pack, the results of the previous "Input Examples" are as follows:

```
"CO CONFIGURE CU 8/1 SN 2 CO INITIALIZE"
```

Unit 8/1 is verified, all files are purged, the disk pack label is written, and pack directory and tables are initialized. The disk pack serial number is 2. The pack is a system resource (unrestricted) disk pack, with a default disk pack name of SYS180, a system interchange code of medium system (MS), and an owner's ID of all blanks. An INITIALIZE is performed on the disk pack specified by the next CU that is input to DISPKV.

```
"CU 8:7/5 SN 11 AC RM ER 1/1/1 ER 2/3/4 PN JAKES OI IVRSUN"
```

Unit 5 is initialized with DLP 7 (the alternate channel) rather than 8 (the primary channel) as a restricted master with a serial number of 11, a disk pack name of JAKES, an owner's ID of IVRSUN, and a system interchange code of MS. The input ER 1/1/1 specifies that cylinder 1, head 1, sector 1 must be relocated. The input ER 2/3/4 specifies that cylinder 2, head 3, sector 4 must be relocated.

```
"CO SINGLE CU 8/4 SN 875 CY 300 ER 300/6/29"
```

A single cylinder IVR is performed for cylinder 300 of the disk pack on channel 8, unit 4. The serial number of the disk pack must be 875. The 29th sector, as accessed by head 6, must be relocated.

```
"CO RELOCATE CU 8/6 SN 39 AD 385472"
```

A single sector RELOCATE of sector 385472 is performed on channel 8, unit 6. The disk pack on channel 8 unit 6 must have a SERIAL NUMBER of 39.

```
"CO CONFIGURE CU 6:5/4 END"
```

Unit 4 is initialized using channel 5 (the alternate channel) rather than channel 6 (the primary channel) and then verified. In addition, all the files are purged, the disk pack label is written, and the disk pack directory and tables are initialized. After the CONFIGURE command is complete, the DISPKV program goes to end-of-job.

Card Input Rules

When you enter the DISPKV parameters on cards, the order is unimportant except that you must use the CO keyword to group a set of data items. Until it encounters the next keyword CO, the program assumes that everything following the keyword CO belongs to the same set.

You can place the input data on several cards, starting in any column. However, the keyword must NOT cross card boundaries. The data associated with a keyword also must NOT cross card boundaries. You can either place an equal sign (=) after the keyword (for instance, CO=INITIALIZE CU=8/3 SN=11 PN=MASTER) or place at least one blank after the keyword.

Figure 3–13 shows a sample DISPKV execution and parameter deck.

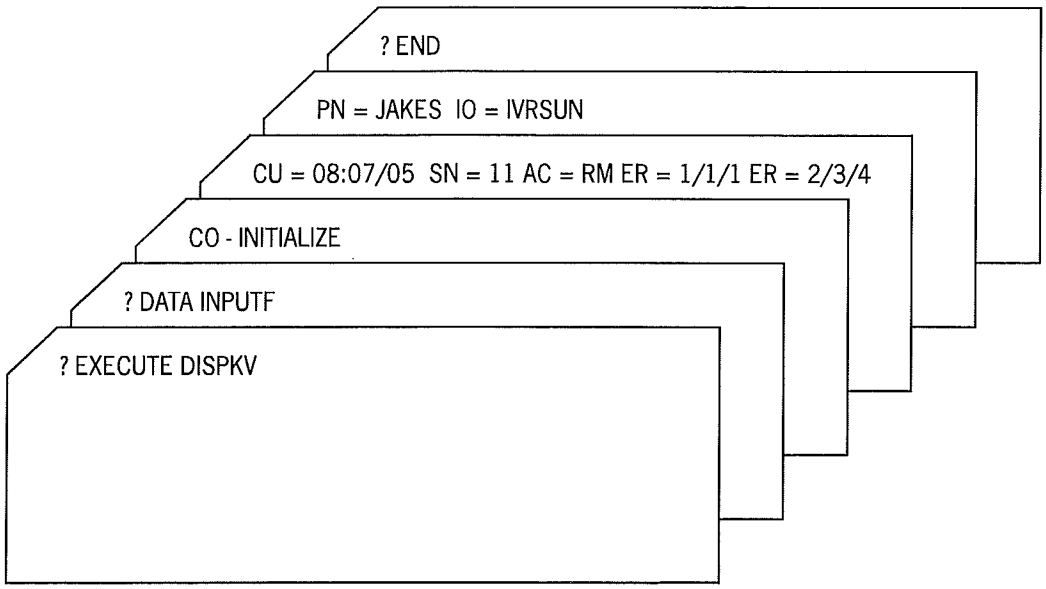


Figure 3–13. Sample DISPKV Executive Deck

DISPKV Messages

The DISPKV program can display messages from any of the following categories:

- **Informational Messages.** These messages indicate the completion of a function and require no operator intervention.
- **Operator Prompts.** These messages prompt you for some type of input.
- **Error Messages.** These messages indicate problems that occur when you are running the DISPKV program.

Informational Messages

The following messages provide information; they do not require a response. However, you may need to act upon the information in the message.

The following message appears on the ODT after a single cylinder IVR if the relocation of a sector was required or if a portion of the cylinder must be removed from the disk available table (with XP system command for 180-byte disk pack or XD system command for 100-byte disk pack):

```
CYLINDER <number> ON CC/U HAS BEEN INITIALIZED AND
THE FOLLOWING ADDRESSES WERE RELOCATED

1 = ccc/hh/ssssss (<sector>) 2 = ccc/hh/ssssss (<sector>)
3 = ccc/hh/ssssss (<sector>) 4 = ccc/hh/ssssss (<sector>)
5 = ccc/hh/ssssss (<sector>)

ALSO, THE FOLLOWING ADDRESSES MUST BE XP'ED
ccc/hh/ssssss THRU ccc/hh/ssssss (<sector> FOR <length>)
```

A sector was corrupted and could not be relocated. The rest of the cylinder could not be verified and, therefore, it must be removed from the available table. On 100-byte disk packs, remove the addresses specified on the ID number to which the disk pack is assigned. On 180-byte disk packs, replace the addresses of the cylinder that were removed before the execution of the DISPKV program (using the RXP system command), and then remove the addresses in the display. The following is an example of this display:

```
CYLINDER 10 ON 14/06 HAS BEEN INITIALIZED AND
THE FOLLOWING ADDRESSES WERE RELOCATED

1 =10/1/60 (4600) 2 =10/1/62 (4602)
3 =10/1/63 (4603) 4 =10/1/82 (4622)
5 =10/4/22 (4832)

ALSO, THE FOLLOWING ADDRESSES MUST BE XP'ED
10/4/33 THRU 10/4/84 (4843 FOR 52)
PACK TYPE 207: DATA FIELD ONLY INITIALIZE
```

A 207 disk pack is being initialized. Only the data portion of every sector is initialized; the addresses are not affected. If any addresses are corrupted, you must perform a single cylinder IVR on the associated cylinder or set a VA 1 = 100000 to override this function if you need a full IVR.

```
PACK TYPE 235: DATA FIELD ONLY INITIALIZE
```

A 235 disk pack is being initialized. Only the data portion of every sector is initialized; the addresses are not affected. If any addresses are corrupted, you must perform a single cylinder IVR on the associated cylinder.

Operator Prompts

When the DISPKV program displays the following prompts, you must provide a response before processing can continue.

The following message appears when you are doing a single cylinder IVR if any of the spares on that cylinder are in use.

```
WANT TO RELOCATE THESE ADDRESSES ON CC/U AGAIN?

1 = cccc/hh/ssssss (<sector>) 2 = cccc/hh/ssssss (<sector>)
3 = cccc/hh/ssssss (<sector>) 4 = cccc/hh/ssssss (<sector>)
5 = cccc/hh/ssssss (<sector>)

ENTER YES, NO, OR # OF ONES TO RELOCATE
```

If you want to relocate all of the sectors again after the initialize function, enter YES. If you do not want to relocate any sectors again, enter NO. If you want selective sectors to be relocated again, enter the number that precedes the equal sign (=) that corresponds to the address to be relocated, as in the following example:

```
WANT TO RELOCATE THESE ADDRESSES ON 14/6 AGAIN?

1 = 10/4/22 (4832) 2 = 10/1/60 (4600)
3 = 10/1/62 (4602) 4 = 10/1/63 (4603)
5 = 10/1/82 (4622)

ENTER YES, NO, OR # OF ONES TO RELOCATE
```

All of the sectors displayed were relocated before the single cylinder IVR was complete. Any or all of the sectors may be relocated again, if desired. If you enter a YES, the program will relocate all five sectors again. If you enter a NO, the program will not relocate any five sectors again. If you enter the numbers 234, the program will relocate sectors 4600, 4602, and 4603 again.

Error Messages

The following error messages may appear on the ODT.

`ACCESS CODE MUST BE BLANK FOR INTERCHANGE PACKS`

The system interchange code specified is IC, and the system access code is not unrestricted. The system access code must be unrestricted on all interchange disk packs.

`COMMAND IS NOT ALLOWED TO THIS UNIT`

This message appears when you enter any of the following commands for a 680, 682, MD4, or MD8 disk pack: INITIALIZE, CONFIGURE, SINGLE, REPORT, or RELOCATE.

`CONTROLLER ERR - RD TAG PRESENT`

A DLP extended result descriptor R/D tag was returned with directions for a controller malfunction, invalid disk Device Dependent Port (DDP) status, controller failure, and so forth.

DLP ERROR R/D RETURNED ON THE DLP

Invalid result descriptor was returned from the DLP. Possible causes are DLP descriptor error, parity error, Device Dependent Port (DDP) parity error, timeout, and so forth.

ERROR CARDS IGNORED, COMMAND IS NOT INITIALIZE

You can specify error addresses only when you use the initialization function.

ERRORS PER HEAD: <head number>=<error count> <head number>=<error count>

This message shows the total number of errors encountered for each read/write head. This message appears only when the total count exceeds 2000 and the DISPKV program terminates.

IOT ERROR CONDITION ON THE DLP

An invalid result descriptor was returned from the IOT. Possible causes are timeout, memory parity error, longitudinal parity error, and so forth.

INVALID INPUT CYLINDER NUMBER

The cylinder number input to the SINGLE command exceeds the maximum number of cylinders for that disk pack.

INVALID LABEL ON MEDIA DESIGNATED

Label on the channel and unit specified is in the incorrect format or cannot be read.

INVALID OWNER'S ID FOR RECONFIGURE

PACK OWNER'S ID on the channel and unit specified does not match the OWNER'S ID input to the DISPKV program. RECONFIGURE must have matching OWNER'S IDs.

INVALID PACK SERIAL NUMBER FOR RECONFIGURE

PACK SERIAL NUMBER on the channel and unit specified does not match the serial number input to the DISPKV program. RECONFIGURE must have matching serial numbers.

INVALID RELOCATE ADDRESS

The sector address input to the RELOCATE command exceeds the maximum addressable sector for that type of disk pack.

INVALID R/D ON READ = <result descriptor>

An attempt to read the spare sectors of a cylinder has failed. The result descriptor specified was returned from this read.

```
INVALID R/D RECEIVED ON ERROR RELOCATE = <result descriptor>
```

Invalid result descriptor was returned on an error sector relocate. Possible causes are parity error, write fault, address error, and so forth.

```
INVALID R/D RECEIVED ON I/O     <result descriptor>
R = RETRY, N = NEXT COMMAND STRING
D = DUMP, NEXT COMMAND STRING
```

An error was found when doing an I/O operation. A response of R will cause the I/O to be retried 10 times or until it is successful. If the I/O fails again after 10 retries the same message is displayed. A response of N will cause the current command to terminate and the DISPKV program will process the next command string. If no further commands are available, the DISPKV program terminates. A response of D is the same as N except that the program will take a memory dump.

```
INVALID R/D RETURNED ON READ UNIT = <result descriptor>
```

An invalid result descriptor was returned from the controller when the program attempted to obtain the firmware revision level and drive type. Possible causes are that the controller is offline, the unit is in maintenance mode, the unit is not present, and so forth.

```
LABEL COULD NOT BE READ
```

The label sector cannot be read after 10 retries.

```
PACK TYPE (IE 235, 206) IS UNDEFINED
```

Pack type returned on read unit ID is not supported by this version of the DISPKV program. The cause is incorrect firmware loaded into the controller or an incorrect version of the DISPKV program.

```
PACK TYPE RETURNED ON TEST OP IS INVALID
```

The disk pack type returned on read unit ID is not supported by this version of the DISPKV program. The cause is incorrect firmware loaded into the controller or an incorrect version of the DISPKV program.

```
PACK'S CAPACITY EXCEEDS THE VERSION 1 LIMIT
APPROXIMATELY 18 GB IS AVAILABLE FOR USE
```

DISPKV has been run against a version 1 diskpack that has a capacity greater than 18 GB. The capacity limit enforced by the MCP for Version 1 diskpacks is 18 GB. For this reason, DISPKV will build the "available" tables for an 18 GB capacity, and the MCP will allocate only 18 GB of the diskpack.

```
REQUEST NOT ALLOWED, ADDRESS <number> FOR <length> NOT XP'ED
```

The address and length specified must be removed before doing a single cylinder IVR (SINGLE). Remove these addresses, using the XP system command on 180-byte disk packs or the XD system command on 100-byte disk packs, and reexecute the DISPKV program.

```
REQUEST NOT ALLOWED, ADDRESS <number> NOT XP'ED
```

The address specified in a RELOCATE command for a 180-byte disk pack was not removed from the disk available table. Remove the appropriate address, using the XP system command, and reexecute the DISPKV program.

```
REQUEST NOT ALLOWED ON 100-byte MEDIA
```

The DISPKV program does not permit a command of RECONFIGURE or RECONFIGUREL on 100-byte disk packs. Tables are not used on these disk packs.

```
SECTOR <number> ON <cc/u> CANNOT BE RELOCATED, LEAVE IT XP'ED
```

The sector input to the DISPKV program cannot be relocated because no spares are available for use, or the available spares are bad. This sector should be removed from the available table (XPed) until full disk pack IVR or single cylinder IVR functions can be performed.

```
SECTOR <number> ON <cc/u> CANNOT BE VERIFIED, <number> IS BAD
```

An attempt failed to verify the cylinder containing the sector address to relocate. The sector specified did not pass the verify and should also be relocated. The original sector input to the DISPKV program was relocated.

Initializing Disk Packs to 100-Byte Mode

To prepare a disk pack in 100-byte mode, perform the following steps:

1. Ensure that the drive has been strapped for 100-byte mode.
2. Declare the spindles using the DISK MCP configuration record. Use the SAVED or RESERVED option so the MCP will not attempt to use the drive for storage. Refer to Volume 1 for more information.
3. Execute DISPKV and initialize the media.
4. Ready the drive with the RY system command.

Section 4
DLPXCO AND DLPXNO—DLP Utility Programs

Overview

DLPXCO and DLPXNO are two utility programs that dump and clear Data Link Processors (DLPs) that hang because of irrecoverable I/O operations.

When the MCP is running, use the DLPXNO program. If the MCP is not up and you are working from a V 300 system, use the floppy disk based DLPXCO program. If the MCP is not up and you are working from a V 500 system, use the DLPXCO program from the hard disk of the maintenance processor.

Both programs read the firmware in the DLP and dump it. Both programs can display the dump information on the screen and print it. The screen display includes the following information:

- The stack, including microcode address
- The microcode word
- The status count of the DLP

Because the DLPXNO program uses the services of the MCP, the program creates a printer backup file of the dump information. You can print this file or copy it to tape for later use. Because the DLPXCO program runs without the support of the MCP, it dumps directly to the printer.

The print file of the DLP dump contains the following information:

- All error messages following the I/O in error
- An I/O trace of console DLP operations
- Logging information from the DLP or connecting module

When the MCP detects a hung DLP, it displays the following message:

```
cc/u <device mnemonic> NO RSLT DESC
<program ID> INV I/O DESC <file ID> <address> X <segment>
<program ID> DS OR DP
```

Both programs display error messages if they encounter problems. A list of these error messages, along with a brief description of each, appears later in this section.

These programs enable you to clear another DLP before resuming normal operations. Note that after these programs are finished, you must reload the firmware.

Executing DLPXCO

The following pages present the procedures you use to execute the floppy disk version of the DLPXCO program at the maintenance processor from a V 300 system and from a V 500 system.

V 300 System

If you are working from a V 300 system, use the following procedure to execute the floppy disk version of the DLPXCO program at the maintenance processor.

1. Press the ON LINE switch on the system cabinet so the word CONSOLE lights up.
2. Insert the floppy disk named CV33nA (where n represents the release number) in the floppy disk drive.
3. If you have set the ODT channel number with the SO command and the ODT option on PANDMV, enter the following command:

   ```
   LOAD SYS DLPXCO; RUN
   ```

   If you have not set the ODT channel number on PANDMV, enter the following command (where nn represents the ODT channel number you will use):

   ```
   LOAD SYS DLPXCO; MEM SYS 34 = nn; RUN 7.
   ```

   You can enter the LOAD, MEM, and RUN commands separately.

4. Press the ON LINE switch on the system cabinet so that the word ON LINE lights up.

   The system prompts you to enter the date, the time, and the printer channel number with the following prompts:

   ```
   ENTER DATE: MMDDYY

   ENTER TIME: HHMM

   ENTER 2 DIGIT PRINTER CHANNEL:
   ```

5. Enter the date (month, day, and year), the time (hour and minute), and the channel number of the printer that will print the dump. If you are not going to print the dump or if the printer is not available, enter FF.

   The system displays the following prompt:

   ```
   ENTER TB CHAN, TEST CHAN AND TYPE AS <BBCCTTT>
   ```

   In this prompt, BB represents the DLP number of the test bus, CC represents the DLP number to be tested and cleared, and TTT represents the DLP type.

6. Enter the test bus channel number, the channel number of the hung DLP, and the DLP type.

   The following are the DLP types:

   - Common front end (CFE) device (this value is the default): HT, MTP, CRD, PCH, or TPR
   - Shared system processor (SSP)
   - Path selection module (PSM)

   For example, you could enter LOAD SYS DLPXCO; RUN

7. When the execution of DLPXCO is complete, reload the firmware required by the DLP.

V 500 System

If you are working from a V 500 system, use the following procedure to execute the DLPXCO program from the hard disk of the maintenance processor.

1. Enter the following command:

   ```
   LOAD PROGRAM DLPXCO SYS Y;
   ```

2. Enter RUN SYS and then press either GO or function key F8 [EVENT].

   The following messages prompt you to enter the date, the time, and the printer channel number:

   ```
   ENTER DATE: MMDDYY

   ENTER TIME: HHMM

   ENTER 2 DIGIT PRINTER CHANNEL:
   ```

3. Enter the date (month, day, and year), the time (hour and minute), and the channel number of the printer that will print the dump. If you are not going to print the dump or if the printer is not available, enter FF.

   The system displays the following prompt:

   ```
   ENTER TB CHAN, TEST CHAN AND TYPE AS <BBCCTTT>
   ```

   In this prompt, BB represents the test bus number, CC represents the DLP number to be tested and cleared, and TTT represents the DLP type.

4. Enter the test bus channel number, the channel number of the hung DLP, and the DLP type.

   The following are the DLP types:

   - Common front end (CFE) device (this value is the default): HT, MTP, CRD, PCH, or TPR
   - Shared system processor (SSP)
   - Path selection module (PSM)

5. When the execution of the DLPXCO program is complete, reload the firmware required by the DLP, and then halt/load the system.

Executing DLPXNO

V 300 System

If you are working from a V 300 system, use the following procedure to execute the DLPXNO program.

1. Enter the following syntax to identify the test bus channel you will use to clear the hung DLP so you can execute the DLPXNO program:

   ```
   UNIT <channel>/1 NST
   ```

   In this syntax, <channel> represents the system console DLP number, as in the following example:

   ```
   UNIT 0/1 NST
   ```

2. Press the transmit (XMT) key after you have identified this channel.

V 500 System

If you are working from a V 500 system, use the following procedure to execute the DLPXNO program.

1. Enter the following syntax to identify the test bus channel you will use to clear the hung DLP so you can execute the DLPXNO program.

   ```
   UNIT 83/0 NST
   ```

   In this syntax, the channel number 83 is the channel number of the V 500 test bus. The DLP channel number is 83.

2. Press the transmit (XMT) key after you have identified the channel.

V 300 and V 500 Systems

From either system, execute the DLPXNO program by performing the following steps.

1. First, enter the following to inhibit the hung DLP so that other programs cannot access it while you are working with it:

   ```
   XC+<DLP channel number>
   ```

2. Next, enter the following to execute DLPXNO:

   ```
   EX DLPXNO (,"ccxx","yyy")
   ```

   In this syntax, cc represents the channel number of the test bus access, xx represents the hung DLP channel number, and yyy represents the DLP type.

   The following are the DLP types:

   - Common front end (CFE) device (this value is the default): HT, MTP, CRD, PCH, or TPR.
   - Shared system processor (SSP)
   - Path selection module (PSM)

Example

```
EX DLPXNO (,"1003","SSP")
```

This example clears an SSP on channel 03, using the test bus interface of the console DLP on channel 10. When the DLP has been cleared, use the following command to return it to the system.

```
XC-<DLP channel number>
```

Then you can load the firmware that the DLP requires.

Error Messages

DLPXCO and DLPXNO can display the following error messages. Recovery procedures are included as needed.

`CONSOLE DLP CANNOT BE CLEARED OR IN BASE WITH MODULE BEING CLEARED`

A Universal Console DLP must be manually cleared. Press the BASE CLEAR button on the DLP base. This procedure is normally performed by a Unisys Customer Service Engineering representative.

`FAILED CONNECTING TO BASE`

The console DLP operation to connect the base was unsuccessful.

`FAILED DISCONNECTING`

The previous console DLP operation to disconnect from the maintenance card of the base being tested was unsuccessful.

`FAILED FINAL CLEAR OF DRIVERS`

The previous console DLP operation to clear the maintenance drivers was unsuccessful.

FAILED INITIAL CLEAR OF DRIVERS

A console DLP operation to clear the maintenance drivers was unsuccessful.

FAILED INITIAL READ

A console DLP operation to read and clear the buffer was unsuccessful.

FAILED LOCAL CLEAR

The previous console DLP operation to clear the DLP was unsuccessful.

FAILED OBTAINING DATA

The previous console DLP operation to collect the logging information in the console DLP buffers was unsuccessful.

FAILED READ OF DATA

The previous console DLP operations to collect the logging information in the console DLP buffers was unsuccessful.

FAILED SETTING OUT

A console DLP operation to specify the unit to be logged and cleared was unsuccessful.

INVALID CLASS OF DLP: <type of DLP>

The type of DLP declared is not correct. Execute the program again, with the correct DLP type.

TEST CHANNEL <declared channel> NOT XCED

The declared channel was not inhibited with the XC command. Verify that the correct channel was entered, inhibit it if necessary, and then execute the program again.

TEST CHANNEL <declared channel> NON NUMERIC

The second two characters identifying the hung DLP were not numeric. Execute the program again with the correct test channel.

UC CHANNEL <console DLP channel number> NON NUMERIC

The first two characters of the console DLP channel number were not numeric. Execute again with the correct channel number.

```
UC CHANNEL <declared channel> NOT UC DLP
```

A check of the declared channel did not return an ID indicating that it is a console DLP. Execute the program again with the correct console DLP channel.

```
UC TEST BUS <UC DLP channel/1> NOT DECLARED
```

The channel specified as the UC DLP did not have unit 1 declared as NST. Verify that the UC DLP channel is declared and that the unit <UC DLP channel/1> is declared as NST; then execute the program again.

These messages require that you reexecute the operation that failed. Contact a Unisys Customer Service Engineering representative if the problem persists.

Section 5
DMPALL—File Conversion Utility Program

Overview

The DMPALL utility program enables you to list, copy, and convert disk, disk pack, tape, and card files. These DMPALL functions are organized as follows:

- **Listing Function.** This function enables you to print the contents of any disk, disk pack, magnetic tape, or punched card file in either an alphanumeric, a digit, or a combined format. You can specify that the printing start or stop at any particular record; you select the record either by record number or by testing the contents of the record.
- **Media Conversion Function.** This function enables you to copy a data file, to change file ID, record length, blocking factor, and parity, and to copy the file to another hardware type. You can specify that this operation start or stop at any particular record; you select the record by record number or by testing the contents of the record. The total buffer size that DMPALL uses to perform this operation must be less than 242K. (The total buffer size is the sum of the input and output record lengths multiplied by their respective blocking factors.)
- **Miscellaneous Functions.** These functions include the following:
  - Printing the contents of a printer backup tape produced by a Unisys A Series or B 5000/B 6000/B 7000 Series system
  - Printing object code disk files in hexadecimal format
  - Listing the names of files that are on a multifile tape, including those produced by the discontinued LOADMP or PACKUP library utility programs
  - Printing the contents of a file that is on a tape produced by the discontinued LOADMP or PACKUP library utility programs

You can initiate the DMPALL program with the PERFORM or PFM command, or with the EXECUTE command. If you use PERFORM or PFM, you should directly enter the parameters described in this section. If you use EXECUTE, you can enter parameters with the AX command, from a card file, or from an executing program.

Using DMPALL to List Files

The DMPALL program lists the contents of files. The following restrictions apply to the DMPALL LIST operation:

- The DMPALL program assumes magnetic tape file (MTP) if you omit the hardware type.
- The DMPALL program accepts only one KEY statement for each execution.
- The options SKIP and STOP take precedence over the options WHEN and UNTIL. The options SKIP, STOP, WHEN and UNTIL take precedence over the KEY option.

When the DMPALL program lists ASCII files, it automatically translates them to EBCDIC unless you use the NT option. When the DMPALL program goes to end-of-job, a count of the records input and the records output appears on the ODT and at the end of the DMPALL listing. The printed listing also includes the DMPALL options selected.

Figure 5–1 shows the syntax for operating the DMPALL program. A discussion of the syntax elements appears after the figure. You can request more than one operation by using the keyword THEN, and following it with syntax for further operations starting with PERFORM or PFM.

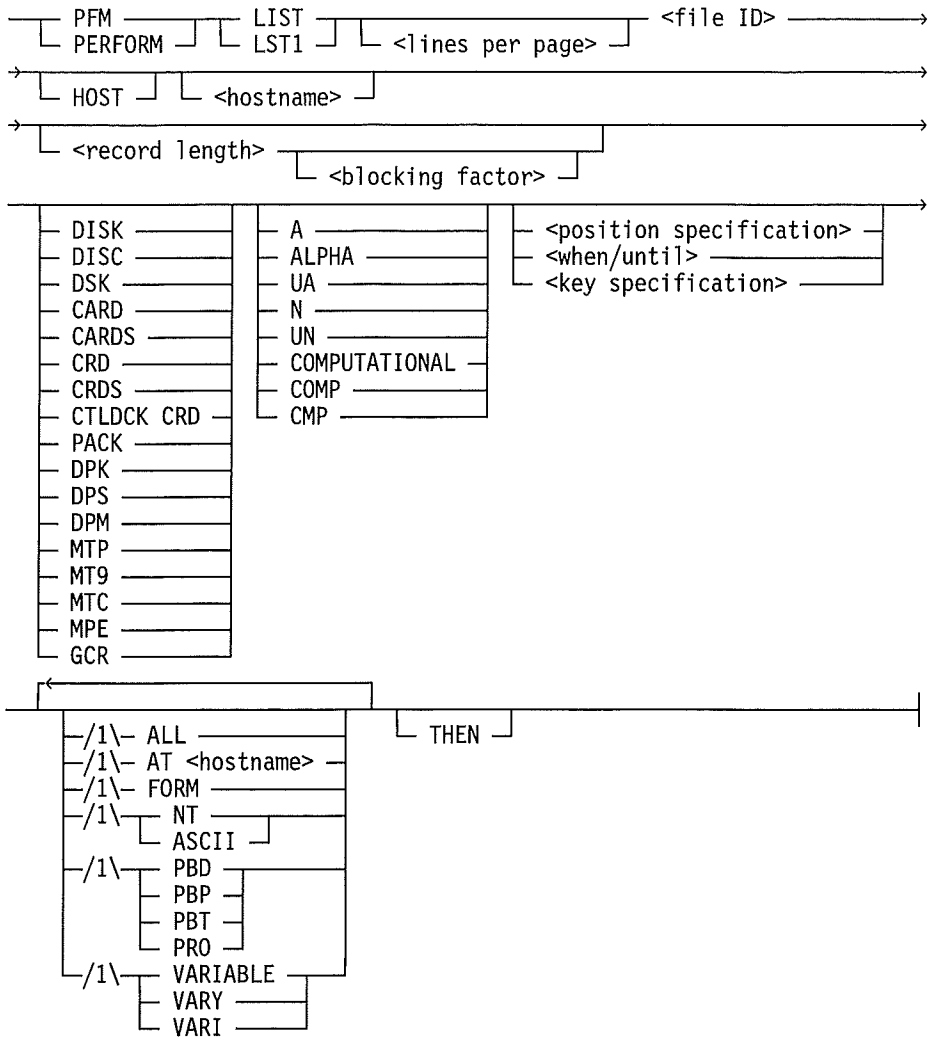
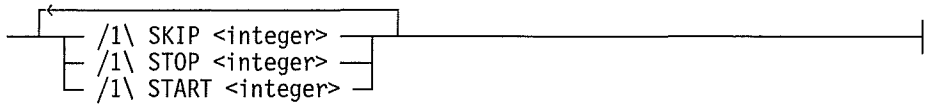
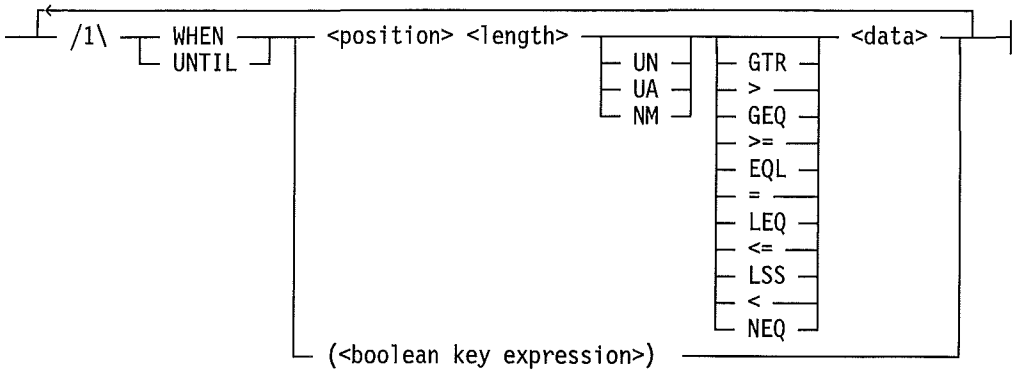


Figure 5–1. DMPALL File Listing Syntax

<position specification>



<when/until>



<key specification>

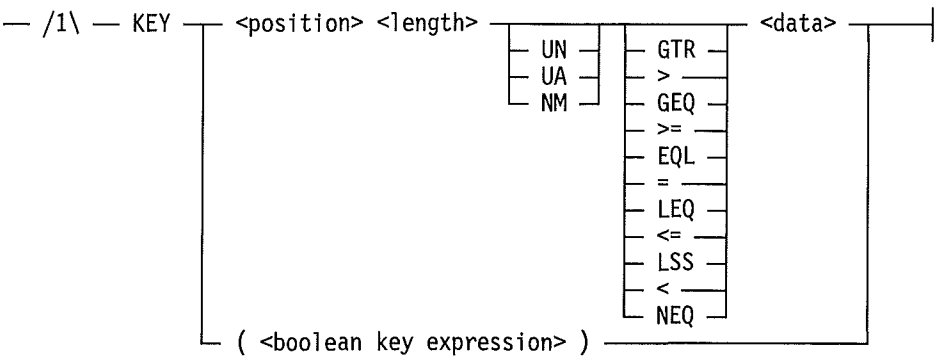


Figure 5–1. DMPALL File Listing Syntax (cont.)

PFM
PERFORM

These options start the DMPALL program. They are equivalent.

LIST
LST1

These options tell DMPALL to list a file, not convert one. LIST provides a double-spaced listing. LST1 provides a single-spaced listing.

<lines per page>

You can use this option to specify the number of lines for each page, excluding heading lines. The 12-punch on a carriage control tape, however, will override any lines per page entry that the 12-punch precedes. The default value for lines per page is 55.

<file ID>

This option is required. It names the file you want listed. For multifile tapes or disk packs, format this parameter as multifile ID/file ID, where the multifile ID is the tape name or the disk pack family name. If the file name contains special characters, surround them with parentheses. The maximum length of the file ID is 8 characters– a 6-character name plus 2 parenthesis characters.

<host name>

Use this option to indicate that the file to be listed resides on a remote host. The output listing is produced on the local host unless you enter the AT host name element. If the host name contains special characters, surround them with parentheses. The host name parameter can consist of 1 to 17 alphanumeric characters plus 2 parenthesis characters if needed.

<record length>

Use this option to format (unblock) tape files or to override the record length given in the disk or disk pack directory for disk or disk pack files. If you omit this option for tape files, the system assumes a record length of the memory size allocated to the DMPALL program minus 6000 digits. If you do not specify a record length for disk or disk pack files, the system uses the record length and blocking factor specified in the disk or disk pack directory. If you do specify the record length, be sure that it is an even number, denoting the length of a logical record in bytes. If you use a blocking factor, always specify a record length first. The maximum record length is 49998 bytes.

<blocking factor>

Use this option to format (unblock) tape records or to override the blocking factor given in the disk or disk pack directory for disk or disk pack files. If you do not specify the value of the blocking factor, the system uses a value of 1, which indicates a single record for each block.

DISK
DISC
DSK

Use these options, which are equivalent, to indicate that the file to be listed is on disk (100-byte-per-sector media).

CARD
CARDS
CRD
CRDS

Use these options, which are equivalent, to indicate that the file to be listed is a punched card file. If there are any control cards in the card file, they will not be listed. A control card is a card with an invalid character, usually a question mark (?), in column one.

CTLDCK CRD

Use this option to indicate that the file to be listed is a punched card file. If there are any control cards in the card file, they **will** be listed. A control card is a card with an invalid character (customarily a question mark, ?) in column 1.

PACK
DPK
DPS
DPM

Use these options to indicate that the file to be listed is a disk pack file. PACK is the default. DPK and DPM indicate that the output file can be spread over more than one spindle of the disk pack family. DPS indicates that the file must be confined to a single disk pack spindle.

MTP
MT9
MTC
MPE
GCR

Use these options to indicate that the file to be listed is a tape file. MTP is the default. MT9 indicates NRZ recording mode. MTC indicates magnetic cartridge tape mode. MPE indicates phase-encoded recording mode. GCR indicates group character recognition recording mode.

A
ALPHA
UA

Use these options, which are equivalent, to list the file in alpha (character) format. Alpha format translates your file into characters so that you can read the text portions. You can enter both this option and the next option (UN or N) to get a listing that includes both alpha and numeric formats, which is the default. The default is a good choice for files such as memory dumps that include both text passages and numeric passages.

N
UN
COMPUTATIONAL
COMP
CMP

Use these options, which are equivalent, to list the file in numeric (digit) format. They print the four-bit hexadecimal characters that make up the file without providing a translation into alphanumeric format. This format can be useful for a numeric data file. You can enter both this option and the preceding option (ALPHA, A, or UA) to get a listing that includes both alpha and numeric formats, which is the default. The default is a good choice for files such as memory dumps that include both text passages and numeric passages.

SKIP <integer>
START <integer>

Use these options, which are equivalent, to create a listing that starts with the record number you enter. Enter any number from 0 to 99999999. If you omit this option or if the integer value equals 1, printing begins with the first record in the file.

STOP <integer>

Use this option to create a listing that stops after the record number you enter. Enter any number from 0 to 99999999.

WHEN

WHEN causes the listing to begin with the first record that meets the conditions you specify. A discussion of the conditional statements appears under the KEY option in this section. You can use the <boolean key expression> to set multiple conditions; all must be met before printing begins.

UNTIL

Use this option to create a listing of the file that begins with the first record that meets the conditions you specify. When the conditional statement becomes true, the DMPALL program stops printing the file and then performs any request given by the THEN instruction. A discussion of the conditional statements appears under the KEY option in this section. You can use the <boolean key expression> to set multiple conditions; all must be met before printing begins.

KEY

Use this option to print selected records. The program selects a record for printing, if a field within it meets the conditions you set.

You can set only one condition in the KEY statement. In the WHEN and UNTIL statements, you can set several conditions, all of which must be met before the statement takes effect.

The following paragraphs describe elements that are used in the WHEN, UNTIL, and KEY statements.

<position>

This option gives the position, within the record, of the field that you want tested. If you use the WHEN, UNTIL, or KEY clause, the <position> is required. Enter this option in digits or bytes, as determined by the UN, UA, or NM entry; the default value is digits. The program determines the test to be applied from the algebraic operators and the data statement. The first digit or byte of the record is digit or byte 1, not 0.

<length>

This option gives the length of the field you want tested. If you use the WHEN, UNTIL, or KEY clause, the <length> is required. Enter this option in digits or bytes, as determined by the UN, UA, or NM element; the default value is digits. The maximum length is 40 digits or 20 bytes.

UN
UA
NM

Use this option to designate the type of data you want tested and to determine the meaning of the position and length options. If you do not specify one of these options, the default value, UN, is used.

UN means 4-bit unsigned numeric data; the position and length options are in digits.

UA means 8-bit unsigned alphanumeric data; the position and length options are in bytes.

NM means 8-bit signed numeric; the position and length options are in bytes.

GTR (or >)
GEQ (or >=)
EQL (or =)
LEQ (or <=)
LSS (or <)
NEQ

These algebraic operators determine how the program is to compare the designated field and the data. The following table shows the operators, their meanings, and their alternate forms.

Greater than	GTR (or >)
Greater than or equal to	GEQ (or >=)
Equal to	EQL (or =)
Less than or equal to	LEQ (or <=)
Less than	LSS (or <)
Not equal to	NEQ

<data>

This option indicates the value that is compared with the value in the specified field. If you use the WHEN, UNTIL, or KEY clause, the <data> is required.

<boolean key expression>

Use this option to enter more than one condition (multiple keys). The conditions are enclosed in parentheses and are separated by the following Boolean operators: NOT, AND, OR. The order of precedence is: NOT, AND, OR.

You can enter a maximum of 20 conditions. Each condition must include the following elements: position, length, and data. In addition, you can specify the type of data and an algebraic operator.

```
WHEN (0 4 UA EQL "TEST" AND 6 2 UN EQL 01)
```

This example has two conditions connected with AND. Records are printed when both conditions are met. A record must have the word TEST in the first four positions and the value 01 beginning at position 6 to satisfy both conditions.

```
KEY (NOT (73 5 UA EQL "ERROR" OR 73 5 UA EQL "DEBUG") )
```

This example has two conditions connected with OR. Records that contain either the word ERROR or the word DEBUG will not be printed. Because NOT is surrounded by an exterior set of parentheses, it applies to either condition in the interior set of parentheses.

ALL

Use this option to print all the files on a multifile tape, beginning with the first file you specify with the file ID option.

AT <host name>

Use this option to specify the name of a remote host on which you want the print file to be created. This option lets you specify only the destination host name for the print file. That is, the input file you want listed is still assumed to reside on the local host unless you enter a host name option after the file ID.

FORM

Use this option to notify the DMPALL program that you want your listing printed on a special forms printer. You specify the printer with the FM system command (refer to Volume 2). You can abbreviate FORM as FM.

NT

Use this option to prevent translation of characters. When you use this option, the DMPALL program will not convert ASCII to EBCDIC or EBCDIC to ASCII.

ASCII

Use this option for proper translation of ASCII data, including unlabeled ASCII tapes. The listing will be in EBCDIC.

PBD

Use this option to direct the printer file to a printer backup disk.

PBP

Use this option to direct the printer file to a printer backup disk pack.

PBT

Use this option to direct the printer file to a printer backup tape.

PRO

Use this option to direct the printer file to a line printer. This is the default, but you can set up the system configuration file to override it.

VARIABLE
VARY
VARI

Use these options, which are equivalent, to list a tape file that has variable record length, variable number of records per block, and variable block size. The first four bytes of each record must contain the length of that record in bytes. You do not need this option for a tape file that has variable record length but is unblocked or has 1 record per block.

THEN

Use this option to enter another string of DMPALL syntax. Follow the keyword THEN with syntax starting with the PFM command. The second string of syntax will be executed after the first string is completed.

Using DMPALL To Convert Files

The syntax that appears in Figures 5–2 and 5–3 shows how to direct the DMPALL program to convert a file from one storage medium to another or to copy a file into another file with different characteristics on the same storage medium.

The following restrictions apply to converting files with the DMPALL program:

- The optional file characteristics for both input and output must be nested. That is, if you specify the blocking factor, you must specify the record length. If you specify the records per area, then you must specify the record length and the blocking factor. If you specify the number of areas, then you must specify all the file characteristics.
- The BIN routine type must not be used with pseudo readers and is prohibited from going to punch backup. In addition, when it is reading binary cards, you can terminate the DMPALL program, using either the QT system command, DS system command, or a binary end card. The output file ID NONE is required when you are converting from binary to binary.
- The control parameter SORT can be specified only in a media conversion operation. The SORT statement will override any specified SKIP, STOP, WHEN, or UNTIL parameters in a media conversion operation. The maximum allowable SORT statements are 10.
- Random files with noncontiguous areas (for example, a file with 5 areas, in which only areas 1 and 5 are allocated) cannot be copied with the DMPALL program.

When the DMPALL program goes to end-of-job, the input record count, output record count, and blocking factor appear on the ODT.

Overview of Syntax for Converting Files with DMPALL

Figure 5–2 is an overview of the syntax for using the DMPALL program to convert files.

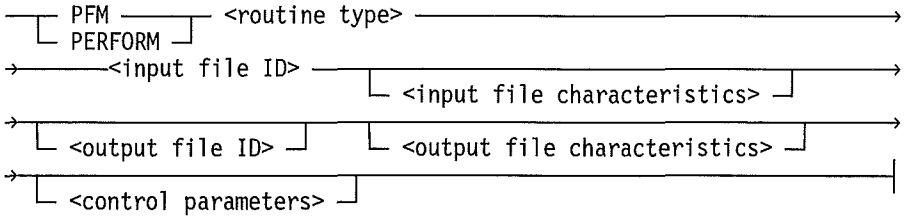
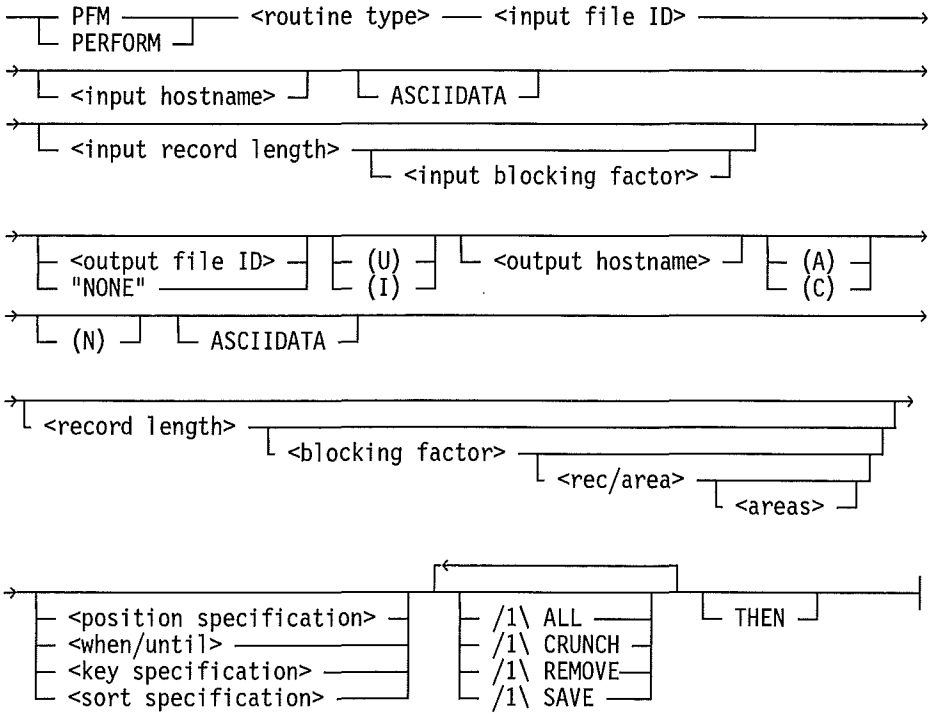


Figure 5–2. DMPALL File Conversion Syntax—Overview

Detailed Syntax for Converting Files with DMPALL

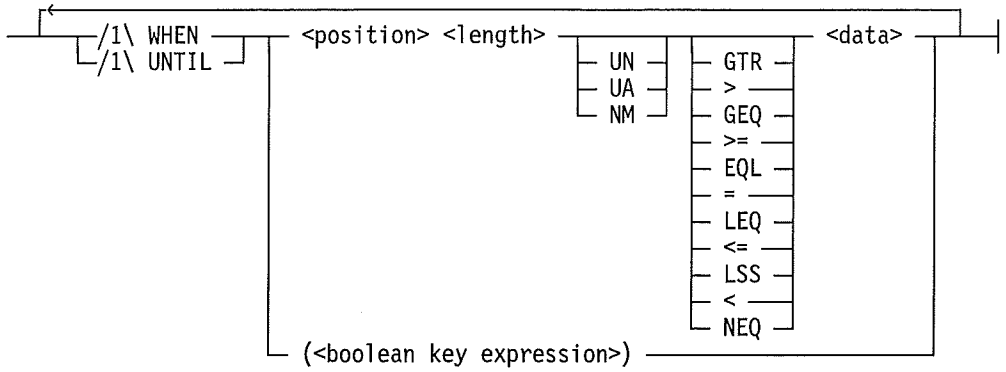
Figure 5–3 shows a detailed syntax for using the DMPALL program to convert media.



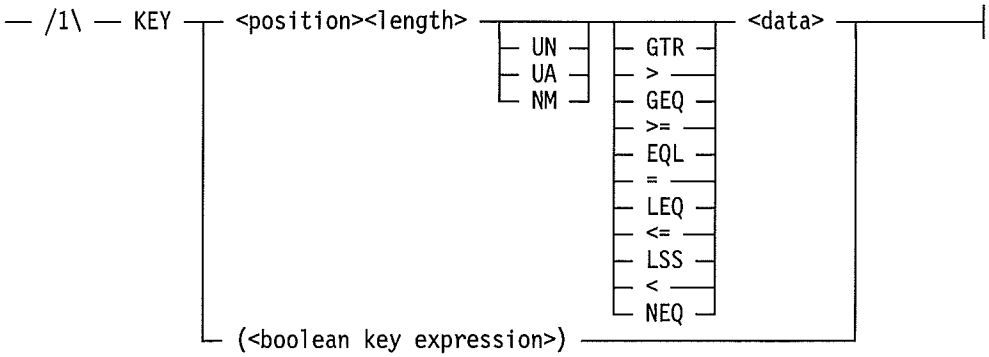
<position specification>

Figure 5–3. DMPALL File Conversion Syntax—Detailed

<when/until>



<key specification>



<sort specification>

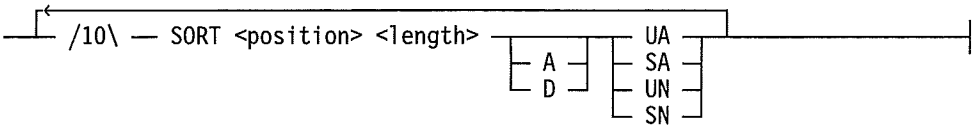


Figure 5–3. DMPALL File Conversion Syntax—Detailed (cont.)

Detailed Media Conversion Options

The following paragraphs describe the options that appear in Figures 5–2 and 5–3. You can select conversion with a looping process by choosing different options for the same parameter. Limits for these processes are specified by the integer associated with the bridge (refer to Figure 5–3). These limits cannot be exceeded. You can request more than one operation by using the THEN keyword and giving the syntax for another operation, starting with the PERFORM or PFM command.

PFM
PERFORM

Use these options, which are equivalent, to start the DMPALL program.

<routine type>

This required option notifies the DMPALL program that you are converting a file, not listing one. Also, this element designates which storage medium is the source and which storage medium is the destination. The routine type consists of a 6-character concatenation of abbreviations for two hardware types. The first three characters specify the input device (where the file you want to convert resides). The last three characters specify the output device (where you want the converted file to be placed). The following table shows the abbreviations you can use for the concatenation:

Binary cards (input or output)	BIN
Card reader (input) or card punch (output)	CRD
Card punch output	CPU
Magnetic tape (NRZ, phase-encoded, or GCR recording mode). This is the default.	MTP
Magnetic tape (NRZ recording mode)	MT9
Magnetic cartridge tape (MTC recording mode)	MTC
Magnetic tape (phase-encoded recording mode)	MPE
Magnetic tape (GCR recording mode)	GCR
Head-per-track-disk or LAK disk pack	DSK or DSC
Disk pack (file can be spread over more than one disk pack spindle)	DPM or DPK
Disk pack (file is confined to a single disk pack spindle)	DPS
Operator Display Terminal	ODT

When you convert a file from disk to disk pack, the concatenation is DSKDPM or DSCDPK.

<input file ID>

This option selects the file that is to be converted. It is required. If you do not want to use an input identifier, you should enter a dummy identifier, and you can then access the file with a UL system command. You must surround special characters with parentheses. The input file ID can be any identifier that is from 1 to 6 characters long (not counting any parentheses). For multifile tapes or disk packs, you can format this entry as <multifile ID/file ID>.

<input file characteristics>

This option describes the file that is to be converted. It is optional; the dafaults are given for each characteristic in the following paragraphs.

<input host name>

This option indicates that the input file resides on a remote host. You can specify a host name preceded by the key word HOST, HOSTNAME, or HN and an optional equal sign.

ASCIIDATA

When used as a qualifier following the <input file ID>, ASCIIDATA indicates that the source file is in ASCII and the destination file is to be translated into EBCDIC. Use this keyword to ensure proper handling of an unlabeled ASCII tape.

<input record length>

This option indicates the record length of the input file, in bytes. The input record length can range from 4 to 49998 bytes. This is optional. For disk and disk pack files, the default record length is the record length defined in the disk or disk pack header. For all other devices, the default is 80 bytes.

<input blocking factor>

This option indicates the blocking factor of the input file. The value can be 1 to 3 digits long. This is optional. For disk and pack files, the default blocking factor is the blocking factor defined in the disk or disk pack header. For all other devices, the default is unblocked files.

<output file ID>

This option indicates the name to be given to the converted file. You must surround any special characters with parentheses. The file name can be from 1 to 6 characters long, not counting any parentheses. A multiple file entry is permitted for tape or disk pack files. This is optional; if you omit it, the output file is given the same name as the input file.

If you want to create an unlabeled file, enter *"NONE"* for the output file ID, including the quotation marks. This applies only to files other than disk or disk pack files.

<output file characteristics>

This option describes the converted file that you want the DMPALL program to produce. It is optional; the defaults are given for each characteristic in the following paragraphs.

(A)
(I)

Use these options to define the file label you want on the output file. If you want to use an ANSI label, enter *(A)*. If you want to use an installation-defined label, enter *(I)*. The use of an ANSI label or an installation-defined label applies only to output files written to devices other than disk or disk pack. This is optional; the default is the Unisys V Series Standard Label.

If you want to create an unlabeled file, enter *"NONE"* for the output file ID. This applies only to files other than disk or disk pack files.

<output host name>

Use this option to indicate that the output file is to be placed on a remote host. You can specify a host name preceded by the keyword HOST, HOSTNAME, or HN.

(A)
(C)

Use these options to determine how the file is assigned on disk or disk pack. Enter *(A)* to have the file assigned by disk or disk pack areas. Enter *(C)* to have the file assigned by disk pack cylinders; this choice is valid only if the destination is disk pack. These are optional; the default is to assign the file by available space.

(N)

Use this option to declare a nonstandard parity. Enter an *(N)* to declare a nonstandard parity for the output file. If you are using a BCL Punch file, you must specify (N).

ASCIIDATA

When used as a qualifier following the <output file ID> or "NONE", the keyword ASCIIDATA indicates that the destination tape is to be translated into ASCII.

<record length>

Use this option to specify the record length of the output file, in bytes. The record length can range from 4 to 49998 bytes. If you omit this option, the program uses the record length of the input file.

<blocking factor>

Use this option to specify the number of records per block for the output file. The value can be 1 to 3 digits long. If you omit this option, the program uses the blocking factor of the input file.

<rec/area>

Use this option to specify the disk or disk pack area size, in records, of the output file. You use this option only for disk and disk pack files. The value can be 1 to 8 digits long. If you omit this option, the output file will have the same number of records per area as the input file. If the input file is not a disk or disk pack file and you do not give a value for output records per area, the program uses the default value of 500 records per area.

<areas>

Use this option to specify the number of disk or disk pack areas assigned to the output file. You use this option only for disk and disk pack files. If you omit this option, the program assumes 20 areas unless the input is from disk or disk pack. If the input file is not a disk or disk pack file and you omit this option, the default value is 20 areas.

<control parameters>

Use these options to choose the records you want to copy from the input file to the output file. Specific examples of control parameters appear under "DMPALL Examples" later in this section. The control parameters include START <integer>, SKIP <integer>, STOP <integer>, WHEN, UNTIL, and KEY.

START <integer>
SKIP <integer>

Use these options, which are equivalent, to start a conversion at the record number you enter, skipping over the previous records. You can enter any number from 0 to 99999999. If you omit this option or if the integer equals 1, conversion begins with the first record in the file.

STOP <integer>

Use this option to stop the conversion after the record number you enter. You can enter any number from 0 to 99999999.

WHEN

Use this option to start the conversion with the first record that meets the conditions you specify. A discussion of the conditional statements appears under the KEY option in this section. You can use the <boolean key expression> to set multiple conditions, all of which must be met before conversion can begin.

UNTIL

Use this option to stop the conversion of the file at the first record that meets the conditions you specify. When the conditional statement becomes true, the DMPALL program stops converting the file and then performs any request given by the THEN instruction. A discussion of the conditional statements appears under the KEY option in this section. You can use the <boolean key expression> to set multiple conditions, all of which must be met before conversion can begin.

KEY

You can use this option to copy selected records from the input file to the output file. A record is selected for conversion if a field within it meets the conditions you set. You select the field to be tested and set the conditions to be met by using the following options.

You can use the <boolean key expression> to set multiple conditions, all of which must be met before conversion can begin.

The following paragraphs describe options that can be used in the WHEN, UNTIL, and KEY statements.

<position>

This option specifies the position, within the record, of the field you want tested. If you use a WHEN, UNTIL, or KEY clause, <position> is required. Enter the position in digits or bytes, as determined by the UN, UA, or NM entry; the default is digits. The algebraic operators and the data statement indicate the test to be applied. The first digit or byte of the record is digit or byte 1, not 0.

<length>

This option to specifies the length of the field you want tested. If you use a WHEN, UNTIL, or KEY clause, <length> is required. Enter the length in digits or bytes, as determined by the UN, UA, or NM element; the default is digits. The maximum length is 40 digits or 20 bytes.

UN
UA
NM

This option designates the type of data you want tested and to determine the meaning of the position and length options. If you do not specify UN, UA, or NM, the program assumes UN.

- UN means 4-bit unsigned numeric data; position and length are in digits.
- UA means 8-bit unsigned alphanumeric data; position and length are in bytes.
- M means 8-bit signed numeric; position and length are in bytes.

GTR (or >)
GEQ (or >=)
EQL (or =)
LEQ (or <=)
LSS (or <) NEQ

Use these required algebraic operators to determine the comparison that the program will perform between the designated field and the data. The following table shows the operators, their meanings, and their alternate forms.

Greater than	GTR (or >)
Greater than or equal to	GEQ (or >=)
Equal to	EQL (or =)
Less than or equal to	LEQ (or <=)
Less than	LSS (or <)
Not equal to	NEQ

<data>

This option designates the value that is compared with the value in the specified field. If you use a WHEN, UNTIL, or KEY clause, <data> is required.

<boolean key expression>

Use this option to enter more than one condition (multiple keys). The conditions are enclosed in parentheses and are separated by the following Boolean operators: NOT, AND, OR. The order of precedence is: NOT, AND, OR.

You can enter a maximum of 20 conditions. Each condition must include the following elements: <position>, <length>, and <data>. In addition, you can specify the type of data and an algebraic operator. The following examples illustrate the use of the <boolean key expression>.

```
WHEN (0 4 UA EQL "TEST" AND 6 2 UN EQL 01)
```

In this example, two conditions are joined by AND. Records are printed when both conditions are met. A record must have the word TEST in the first four positions and the value 01 beginning at position 6 to satisfy both conditions.

```
KEY (NOT (73 5 UA EQL "ERROR" OR 73 5 UA EQL "DEBUG") )
```

In this example, two conditions are joined by OR. If records contain either the word ERROR or the word DEBUG they will not be printed. Because NOT is surrounded by an exterior set of parentheses, it applies to either condition in the interior set of parentheses.

SORT

This option causes the DMPALL program to call the SORT. intrinsic program with the parameters you specify to sort the input file records into the output file. You can enter multiple SORT statements to designate multiple sort keys. The first key you enter is the most significant. You can enter a maximum of 10 SORT statements.

The final sequence of the sorted output file is based on the sort parameters. The following paragraphs describe the sort parameters, which include position, length, ASCENDING, A, DESCENDING, D, UA, SA, UN, and SN.

The SORT. intrinsic program is described in Section 21.

The following paragraphs describe options that can be used in the SORT statement.

<position>

This option indicates the position in the record of the field that is to be used as a sort key. You enter this option in digits if you specify UN or SN as the data type or in bytes if you specify UA or SA as the data type.

<length>

Use this required option to specify the length of the field that is to be used as a sort key. You enter this option in digits if you specify UN or SN as the data type or in bytes if you specify UA or SA as the data type.

A (ASCENDING)
D (DESCENDING)

Use this option, which is optional, to determine the sequence of records in the sorted output file. The abbreviation A stands for ASCENDING; the abbreviation D stands for DESCENDING. The default value is DESCENDING.

UA
SA
UN
SN

Use this option to specify the data type of the sort key(s). The default value is UN. The following table shows the values for each data type:

8-bit unsigned alphanumeric	UA
8-bit signed alphanumeric	SA
4-bit unsigned numeric computational	UN
4-bit signed numeric computational	SN

The following options apply to the entire conversion operation.

ALL

Use this option during a tape-to-tape operation to copy all files on a multifile tape. Copying begins with the file you identified in the input file ID.

CRUNCH

Use this option to specify that the output disk or disk pack file is to be closed with CRUNCH. This means that the file is condensed. All assigned but unused areas are discarded, and the records per area value is reduced to the actual records used.

REMOVE

Use this option to close disk and disk pack output files. When you do so, any existing file that has the same name as the output file on the destination medium is removed automatically.

SAVE

Use this option to cause a SAVE WITH LOCK operation on the output file if the DMPALL program is terminated abnormally.

THEN

Use this option to enter another string of DMPALL syntax. After the keyword THEN, enter syntax starting with the PFM command. This syntax will be executed following the completion of the first syntax.

Using Miscellaneous DMPALL Functions

The following paragraphs describe miscellaneous functions that you can perform with the DMPALL program. Several features are included in the program for operator or programmer convenience.

Listing LOADMP and PACKUP Library Tape Files

The PERFORM LIBLST function causes the DMPALL program to print the contents of a file from a tape produced by the LOADMP or PACKUP programs. (These programs are no longer supported by Unisys.) The format of the file you want to list must be the same as that of a printer backup disk file. The syntax for printing a file from these library tapes is as follows:

```
PFM LIBLST <tape ID> <file ID>
```

Example

```
PFM LIBLST LIBT AAAA
```

Listing Object Code Files from Disk

The PERFORM LIST DSK function causes the DMPALL program to produce a hexadecimal listing of the object code disk file specified by the file ID. This function also formats the segment dictionary and indicates the location on disk or diskpack of the beginning of each logical program segment. The syntax for printing an object code file is as follows:

```
PFM LIST <file ID> DSK CODE
```

Example

```
PFM LIST AAAA DSK CODE
```

Listing A Series Printer Backup Tapes

The PERFORM PB67 function causes the DMPALL program to print the contents of an A Series or B 5000/B 6000/B 7000 printer backup tape. The tape must be labeled BACKUP. The correct channel/unit can be designated by an IL (Ignore Label) system command. Refer to Volume 2 for more information. The syntax for printing a file from an A Series printer backup tape is as follows:

```
PFM PB67
```

Displaying Tape Directories

The SEARCH function lists the names of the files on a multifile tape. This list is primarily for use with multifile tapes created by programs running on V Series systems. The DMPALL program locates a tape that has a tape name (multifile ID) that matches the tape

ID you enter. Then, the DMPALL program scans the tape. The DMPALL program displays all the file names on the ODT and lists information for each file. The syntax for listing the file names from a multifile tape is as follows:

```
PFM LIST <tape-id> SEARCH
```

or

```
PFM LIST <tape-id> SEA
```

Example

```
PFM LIST MYTAPE SEARCH
```

Listing LOADMP and PACKUP Library Tape Directories

The PERFORM PD function causes the DMPALL program to print a directory listing of a library tape produced by the LOADMP or PACKUP programs. (LOADMP and PACKUP are no longer supported by Unisys.) The syntax for printing a directory listing of these tapes is as follows:

```
PFM PD <tape-id>
```

or

```
PFM PD <channel number/unit number>
```

Example

```
PFM PD LIBT PFM PD 6/2
```

Using the ZIP Mechanism

ZIP is a programmatic interface for MCP system commands. The keyword ZIP followed by the name of a program causes that program to be started when the DMPALL program goes to end-of-job. Both of the following examples cause the program named PROGA to be scheduled for execution after the DMPALL program finishes.

Examples

```
PFM LIST AAAA ZIP PROGA
```

In this example, the DMPALL program lists the file called AAAA. When the DMPALL program ends, the program called PROGA begins.

```
1OAX DSKMTP AAAA BBBB ZIP PROGA
```

In this example, the DMPALL program lists the file called AAAA on the tape called BBBB. When the DMPALL program ends, the program called PROGA begins.

Starting DMPALL with the EXECUTE Command

Start the DMPALL program with the EXECUTE command instead of the PERFORM command. You can enter EXECUTE from an ODT, a remote terminal, a card file, or an executing program. When you start the DMPALL program in this way, you give it parameters with the AX (Accept) command. Refer to Volume 2 for more information about commands.

If you start the DMPALL program with the PERFORM command, you can use all of the options presented under "Using DMPALL to Convert Files" earlier in this section. In addition, you can use a VALUE clause to access features that enable you to give commands to the DMPALL program from a card file or from an executing program. These VALUE clauses are discussed in this section.

How to Start DMPALL with the EXECUTE Command

To start DMPALL with the EXECUTE command, proceed as follows:

1. At the ODT or a remote terminal, enter this command:

   ```
   EXECUTE DMPALL
   ```

   The DMPALL program displays the following prompt:

   ```
   DMPALL = <mix no> ACCEPT
   ```

2. Enter an AX command as shown here:

   ```
   <mix no> AX <options>
   ```

   The mix no is the mix number of the DMPALL program as displayed in the Accept message. The options are the DMPALL parameters are shown in Figures 5–1 and 5–3.

Example

```
EX DMPALL BOJ DMPALL=002 (MCP) 08:10  DMPALL=002 ACCEPT
2AX PFM LIST AAA STOP 100
```

To start DMPALL from a card file or an executing program, refer to the VALUE clauses in this section.

VALUE 0 = 100000

The EXECUTE DMPALL VALUE 0 = 100000 statement enables you to enter one PERFORM or PFM command to the DMPALL program from a card reader. You can enter the EXECUTE instruction from the ODT or from a card deck. When you use this statement, the DMPALL program searches for a card file with the file name of DMPALL and executes the instruction it contains. The first card in the file must say ?DATA DMPALL. The instructions can occur in any column on the cards. If there is more than one PERFORM or PFM instruction in the card deck, the program uses only the last one.

Example

```
?EXECUTE DMPALL VALUE 0 = 100000 ?DATA DMPALL
PFM DSKMTP AAAA 5000 7 350 BBBB 5000 1 ?END
```

This example is of a card deck that starts the DMPALL program and copies a file from disk to tape (DSKMTP) with input (AAAA) and output (BBBB) having record lengths of 5000 bytes. The input disk file (AAAA) has 350 records per area with a blocking factor of 7. The output file (BBBB) is unblocked.

VALUE 0 = 000001

The EXECUTE DMPALL VALUE 0 = 000001 statement enables you to enter more than one PERFORM or PFM command to the DMPALL program at one time from a card reader. You can enter the EXECUTE statement from the ODT or from a card deck. When you use this statement, the DMPALL program searches for a card file called DMPALL and then executes the instructions in that file. The first card in the file must say ?DATA DMPALL. The instructions can occur in any column position on the cards. If you use parentheses or quotation marks, the left and right characters must be on the same card.

Example

```
?EXECUTE DMPALL VALUE 0 = 000001 ?DATA DMPALL PFM DSKMTP AAAA AFILE/AAAA PFM
DSKMTP BBBB AFILE/BBBB PFM DSKMTP CCCC AFILE/CCCC ?END
```

This example shows syntax that starts DMPALL and creates a multifile tape (AFILE) with the files AAAA, BBBB, and CCCC being copied on the tape (AFILE). The output (tape) files will have the same record lengths and blocking factors as the input (disk) files.

VALUE 0 = 001000

The EXECUTE DMPALL VALUE 0 = 001000 statement enables the DMPALL program to receive data from another program by way of a FILL statement when the language used provides this capability. The FILL statement invokes the Core-to-Core interprogram communication function to get the DMPALL parameters.

The DMPALL program does a global FILL from any program. The program sending the DMPALL parameters must send them to the program named *DMPALL*. You must provide the program to send the DMPALL parameters. The DMPALL parameters can start in any column, but can be no longer than 60 characters.

Example

```
WORKING-STORAGE SECTION.
77 DMPALL-EXECUTE PIC X(72)     VALUE "EX DMPALL VA 0 = 1000. ".
77 SPECS PIC X(60)              VALUE "PFM LIST MSTRFL DSK STOP 100. "
.   .   .
PROCEDURE DIVISION.
START-DMPALL.
ZIP DMPALL-EXECUTE.
FILL SPECS INTO "DMPALL".
```

In this example, the ZIP statement will start the DMPALL program and inform it that the specifications will come from Core-to-Core. The FILL statement will then supply the specifications.

Encountering a Parity Error

When the DMPALL program encounters a parity error on an input tape, the system displays a warning message indicating the error and then requests that you enter an AX or DS (Discontinue) command.

Examples

```
==> RP 6/4  06/4 MPE PURGED  06/1 MTP PAR ERR
DMPALL=14 DROPPED BLOCKING # 16  DMPALL=14 AX or DS
```

These examples show the messages produced at the ODT by a parity error.

When the system encounters a parity error, you can perform one of the following operations:

- Terminate the DMPALL program by entering the following:

  ```
  <mix no> DS
  ```

  The <mix no> value is the mix number of the DMPALL program, as displayed in the request to accept or discontinue.

  Example

  ```
  14 DS
  ```

- Continue processing and ignore this parity error and all subsequent parity errors in this execution of the DMPALL program by entering the following:

  ```
  <mix no> AX IGNORE
  ```

  The IGNORE clause can be abbreviated as I, IG, IGN, IGNO, or IGNOR.

  Example

  ```
  14 AX IGNORE 14 AX IGN
  ```

- Continue processing and ignore this specific parity error by entering the following:

  ```
  <mix no> AX XXXXX
  ```

  If you use any word (customarily XXXXX) except IGNORE or its abbreviations (I, IG, IGN, IGNO, or IGNOR), the DMPALL program ignores only the first parity error. Any subsequent parity errors cause the DMPALL program to stop. When the program stops, it issues a warning message and an AX or DS option appears on the ODT. At this point, you can enter any response, including IGNORE, and the DMPALL program continues processing.

Example

```
14 AX XXXXX
```

DMPALL Examples

```
PFM LIST TAP1 ALL PFM LIST TAP1/CCCC ALL
```

The first command in this example causes all files on TAP1 to print. The second command causes all files on TAP1 to print, beginning with file CCCC.

```
PFM LIST AAAA AT MCPSystemB
```

This command causes the AAAA file to print on another system (MCPSystemB).

```
PFM LIST AAAA FORM
```

This command notifies the DMPALL program that special forms are to be printed on a specific printer through the FM system command.

```
PFM LIST AAAA HOST
```

This command indicates that the file AAAA resides on a remote host.

```
PFM LIST AAAA KEY 10 2 UA EQL AB
```

This command causes only those records that are in file AAAA and that have AB in bytes 10–11 to print. The contents of the comparison are based in the KEY option specified by position 10, for a field length of 2, and for the specified data AB. The data type tested is defined as UA. The record is printed if the EQL condition operator is matched.

```
PFM LIST AAAA
```

This command causes records from the AAAA file to be printed in double-spaced format.

```
PFM LST1 AAAA
```

This command causes records from the AAAA file to be printed in single-spaced format.

```
PFM LIST AAAA NT
```

This command prints the contents of a magnetic tape with no translation (NT).

```
PFM LIST AAAA PBD
```

This command prints the file AAAA to a printer backup disk (PBD).

```
PFM LIST AAAA PBP
```

This command prints the file AAAA to a printer backup disk pack (PBP).

```
PFM LIST AAAA PBT
```

This command prints the file AAAA to a printer backup tape (PBT).

```
PFM LIST AAAA PRO
```

This command prints the file AAAA to a line printer (PRO).

```
PFM LIST AAAA SKIP 100
```

This command causes printing to begin with the record number denoted by the integer 100.

```
PFM LIST AAAA START 100
```

This command causes printing to begin with the record number denoted by integer 100. If you specify integer 0, printing begins with the first record in the file.

```
PFM LIST AAAA STOP 30
```

This STOP command causes the first 30 records (records 1–30) of the AAAA file to be printed.

```
PFM LIST AAAA DSK STOP 30 THEN PFM DSKMT9 AAAA MASTER 100 5
```

This command lists the first 30 records of the disk file AAAA. Then the file AAAA is written to a magnetic tape with a file ID of MASTER. According to this command, the output tape MASTER has a record length of 100 bytes and a blocking factor of 5.

```
PFM LST1 AAAA VARIABLE A
```

This command produces a single-spaced listing (LST1), in alpha format (A), of the data on a variable-length record tape labeled AAAA.

```
PFM LIST AAAA DSK WHEN 10 2 UA EQL AB
```

This command causes printing to begin with the record that is first in the AAAA file and that matches the WHEN parameter. The contents of comparison are based in the WHEN option specified by position 10, for a field length of 2, and for the specified data AB. The data type tested is UA. The record is printed if the EQL condition operator is matched.

```
PFM LIST AAAA DSK UNTIL 10 2 UA EQL AB THEN PFM DSKMT9 AAAA BBBB 100 5
```

This command indicates that printing begins immediately, but stops when the first record in the AAAA file matches the UNTIL option. The contents of comparison are based in the UNTIL option specified by position 10, for a field length of 2, and the specified data field AB. The data type tested is UA. The DMPALL program stops the

print process when the EQL condition operator is met; then it performs the next function.

```
PFM MTPMTP AAAA BBBB ALL
```

This command initiates the copying of all files on AAAA.

```
PFM DSKDSK AAAA BBBB CRUNCH
```

This command closes and condenses the file BBBB by indicating CRUNCH.

```
PFM DSKDSK AAAA BBBB KEY 10 2 UA EQL AB
```

This command indicates that only those records that are in file AAAA and that have AB in bytes 10–11 are selected. The contents of comparison are based in the KEY option specified by position 10, for a field length of 2, and the specified data field AB. The data type tested as UA. The record is copied if the EQL condition operator is matched.

```
PFM DSKDSK AAAA BBBB REMOVE
```

This command copies the file AAAA under the new name BBBB. Old copies of BBBB are removed automatically.

```
PFM DSKDSK AAAA BBBB SAVE
```

In this example, the option SAVE closes the file BBBB with lock if DMPALL terminates abnormally.

```
PFM DSKDSK AAAA BBBB SKIP 100
```

In this example, SKIP causes the program to select records beginning at record 100 in the AAAA file.

```
PFM DSKDSK AAAA BBBB SORT 1 7 D UA
```

In this example, PFM causes a disk file named AAAA to be sorted in descending sequence. The SORT key has unsigned alphanumeric data, a position in the first byte of the record, and a length of 7 bytes. The sorted output disk file is named BBBB.

```
PFM DSKDSK AAAA BBBB START 100
```

In this example, START causes the selection of records in the AAAA file to begin at record 100.

```
PFM DSKDSK AAAA BBBB STOP 500
```

In this example, STOP causes the first 500 records of the AAAA file to be copied.

```
PFM DSKDSK AAAA BBBB STOP 100 THEN PFM DSKMTP AAAA BBBB MASTER 100 5
```

In this example, the first 100 records of the disk file AAAA are copied to disk, using the same file name. Then the AAAA file is copied to a magnetic tape with an output file ID named MASTER. The output tape has a record length of 100 bytes and a blocking factor of 5.

```
PFM DSKDSK AAAA BBBB WHEN 10 2 UA EQL AB
```

In this example, copying begins with the first record that is in the AAAA file and that matches the WHEN option. The contents of comparison are based in the WHEN option specified by position 10, for a field length of 2, and the specified data field AB. The data type tested is UA. Copying begins when the EQL condition operator is matched.

```
PFM DSKDSK AAAA BBBB UNTIL 10 2 UA EQL AB THEN PFM DSKMTP AAAA BBBB 100 5
```

In this example, copying begins immediately, but stops when the first record in the AAAA file matches the UNTIL option. The contents of comparison are based in the UNTIL option specified by position 10, for a field length of 2, and the specified data field AB. The data type tested is UA. The DMPALL program stops the copy when the EQL condition is met; then the program performs any new request. In this example, the THEN function is executed.

Section 6
DMPANL—MCP Memory Dump Analysis Program

Overview

The DMPANL program reads an MCP memory dump from disk, disk pack, or tape and produces a printer backup file containing a formatted analysis of the dump. The internal name of the printer backup file is DUMP/MEMORY.

Requirements for Dump Analysis

The following requirements must be met before you can use the DMPANL program:

- Include a USE record specifying the dump file location in the cold-start deck (or the system configuration file), as in the following example:

```
USE DUMP DISK
```

- If you do not include a USE DUMP record or a USE DUMP TAPE record, use the DMPMEM program to obtain a memory dump. Refer to Volume 1 for information about the USE record with the DUMP option.
- Use the standard MCP LINKER template to link the MCP. The DMPANL program requires that all link decks contain the same number of modules in the same relative order.
- Be sure that the version of the MCP MID that you use to link an MCP is compatible with the versions of the DMPANL and the DMPOUT programs.
- Do not patch the XM firmware to halt on hardware calls. Such patches prevent entry to the fault handler that produces the dump.

Starting DMPANL Under MCP/VS 3.n

To analyze an MCP/VS 3.n dump on a machine that is running MCP/VS 3.n, use the PM 1 system command to initiate the DMPANL program.

By stringing the commands after the PM 1, you can enter the DMPANL command syntax when you enter the PM 1 command, as in the following example:

```
PM 1 ALL
```

You can also enter the command followed by an asterisk. The asterisk makes the DMPANL program wait for commands in the form of a mix number AX (Accept) command. If you use the AX command, you must terminate the string of commands with the word END, as in the following example:

Enter the following:

```
PM 1 *
```

The system responds with the following message:

```
DMPANL=018 ENTER <DUMP SPECS> or HELP or END:
 ** DMPANL=018 ACCEPT
```

You respond by entering the following:

```
18 AX ALL END
```

Commands for the DMPANL program are additive; that is, if you request a given table anywhere in the DMPANL commands, the program will print it.

DMPANL Command Syntax

Because of the ever-increasing size of the MCP tables, limiting the information you want to print will become more and more important. The command syntax enables you to specify what is to be printed, down to the level of the specific tables. The syntax appears later in this section.

- If you enter the PM 1 command with no parameters, the results depend on how you set up the USE record with the DUMP option in the system configuration file:
  - If the USE record with the DUMP option specifies DISK or PACK, the DMPANL program produces a brief summary listing and copies the entire memory dump to a tape.
  - If the USE record with the DUMP option specifies TAPE, the DMPANL program produces an error message because no memory dump file is available.
- If you enter an asterisk:
  - The DMPANL program displays the following message:

    ```
    ENTER <DUMP SPECS> OR HELP OR END
    ```

  - The DMPANL program issues an ACCEPT and awaits input.
  - You respond as follows:

    ```
    <mix no> AX <dmpanl parameters>
    ```

- If you use the asterisk (*) and enter AX commands, then you can enter multiline input.

- Enter *END* to finish the DMPANL command syntax.
- Enter *<mix no>* AX HELP to get a list of DMPANL parameters. Then the DMPANL program awaits input.

If the system detects a syntax error in the parameters, the system displays an error message and ignores all parameters following the one in error. Then, the DMPANL program continues to await input.

Figure 6–1 shows the syntax for the PM command for MCP dumps.

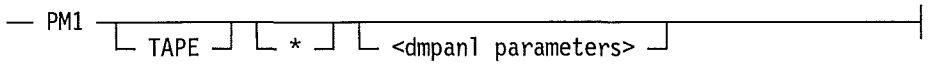


Figure 6–1. PM Command Syntax for MCP Dumps

TAPE

This keyword indicates that the source of the memory dump is tape. The tape could have been created with the DMPMEM or DMPANL program as described in this section. If you do not enter any other parameters, the DMPANL program produces a brief summary listing.

*

If you enter an asterisk, the DMPANL program issues an ACCEPT and awaits input. Enter multiline input in this manner.

DMPANL Parameters

Figure 6–2 shows the parameters for the DMPANL command.

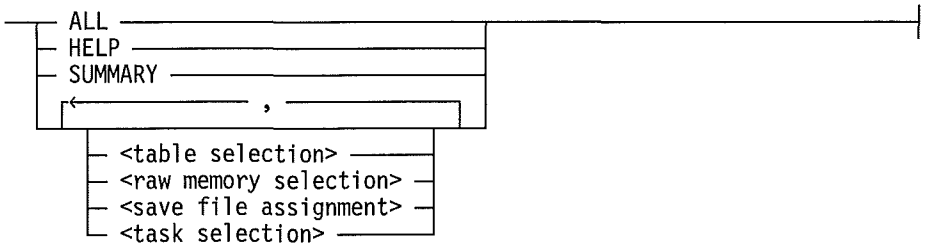


Figure 6–2. DMPANL Command Parameters

ALL

This parameter prints all the loaded MCP, the system tables, and all other allocated memory.

In most cases, the listing you produce with the ALL parameter will exceed the maximum size of the disk or disk pack printer backup file. Use the file-equate clause to send the PRINT file to tape.

HELP

This parameter produces a list of the DMPANL parameters. You can enter it only after an asterisk (*). After producing the list, the DMPANL program continues to await input.

SUMMARY

This parameter prints a brief summary of the dump, including the current task information, the function path, the MCP stack frame of the current task, the MCP object patch history list, and the patch history lock value.

DMPANL Table Selection

Figure 6–3 shows the options for DMPANL table selection.

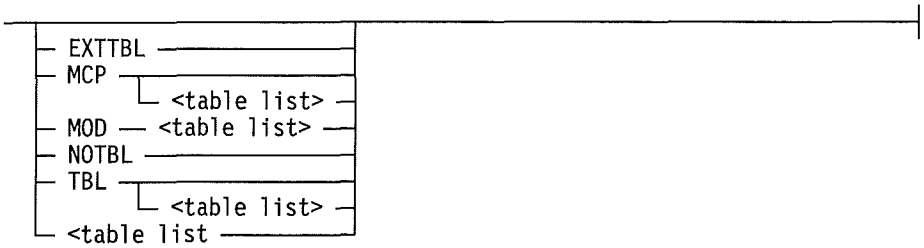


Figure 6–3. DMPANL Table Selection Options

EXTTBL

This option prints the system tables and all the task MCP tables, such as the soft IOATs, the incore DFHDRs, and the address blocks. This option does not produce a raw memory dump.

MCP

This option produces a system table breakout and a raw dump of the MCP and overlay area.

In most cases, the listing you produce with the MCP option will exceed the maximum disk or disk pack printer backup file size. Use the file equate clause to send the PRINT file to tape.

MOD

This option prints the system tables, the memory from 0 KD to the top of the overlay area, any MCP modules that were loaded, and any independent runners that were executing.

In most cases, the listing you produce with the MOD option will exceed the maximum disk or disk pack printer backup file size. Use the file equate clause to send the PRINT file to tape.

NOTBL

This option overrides the default that prints the system tables when you request specific MCP tables.

TBL

This option prints the system tables, but does not print a raw memory dump.

Table List

The following paragraphs describe the data structures (tables) of the internal operating system that the DMPANL program can analyze.

AVLQUE

This option prints all of the I/O queue elements from the available queue list.

CHAN <channel #>[<channel #>]:ALL:

This option prints out all of the I/O activity for the channel number(s) you specify. The printout includes the channel table entry and device table entry for all units on the channel; all I/O, service, and available elements queued on the channel, on any unit on the channel, or on any exchange associated with the channel; and a table that shows the address and position of the queue elements in the IOQUE link list.

Specify as many channel numbers as you want (separated by blank characters), or specify ALL to get a printout of the I/O activity of all of the channel numbers existing in the channel table.

CHTBL

This option prints the channel table for all the channels in the system.

CMPLX

This option prints information about tasks waiting in complex wait.

DCP

This option prints all of the information relating to the data communication processors. This information includes the DCP station table, and the DCP and MCS tables and their entries and buffers.

DEVTBL

This option prints the information from the device table for all of the devices connected to the system.

DMS

This option prints information from two internal DMSII data structures, the database program (DBP) table and the database user program control table.

EU

This option prints the EU table and the disk subsystem table.

FILE

This option prints the following information for every open file:

- The FIB
- The IOAT
- The external FIB
- The incore DFHDR
- The address blocks

This option also prints the following for every open PORT file:

- The Port Attribute Structure
- The Subport Directory
- The Subport Attribute Structures

GLOBAL

This option prints the global code and overlay segment.

HLPARMS

This option prints the halt/load parameters.

IOAT

This option prints the IOAT table.

IOQUE

This option lists all of the queue elements that are in the I/O queue linked list.

JMR

This option prints internal data structures that the Job Manager uses.

MAST

This option prints the memory area status table and provides the location where memory is available for use and is in use on the system.

MAT

This option prints the memory area tables for every task and MCP function.

MIX

This option prints the Mix table.

MSGPOOL

This option lists all of the information that is linked to the message activity in the system. This option invokes the MSGSND, MSGREC, and MSGTWT options.

MSGSND

This option prints all of the information from the send list of the message module.

MSGREC

This option prints all the information from the receive list of the message module.

MSGTWT

This option prints all of the information from the task wait table. The information is printed in ascending order by task number.

PORTS

This option prints information from the port global data memory area and the candidate list memory area.

The port global data memory area contains entries for active ports on the system. Each entry contains attributes shared by all subports connected to the port, including an event directory.

The candidate list memory area contains entries for subports the system is trying to open. An entry contains attributes needed to match subports, control state, and status information.

SEC

This option prints the security user table.

SEG

This option prints the segment dictionary table.

SERVQUE

This option lists all of the elements that are in the service queue linked list.

QUEUES

This option lists all of the I/O queue elements that are in memory area 3. Use this option to track I/O queue elements that are no longer linked in the IOQUE linked list but still exist (in original form) in the memory area.

SLOG

This option prints the last hundred records of the ODT log.

SNAP

This option prints a formatted listing of the SNAP picture contained in the memory dump file. SNAP pictures are inserted into memory under certain program or system failure conditions. The contents of a SNAP picture differs depending on the type of V Series system (V 300, V 500, and so forth).

SUBPORTIO

This option prints the contents of the shared subport memory area (SSMA). There is one SSMA for each pair of open subports on the system. An area contains the input and output data queues and attributes shared by both connected subports.

TRAK

This option prints the TRAK buffer.

TS

This option prints the timesharing system information block and the shared area table.

V 500

This option can be used to analyze a dump file from a V 500 on any system.

VCS

This option prints the internal data structures used by the V Series Communication System (VCS).

WAIT

This option prints the information about every task waiting on the system, including the locks it owns and the locks it is attempting to get.

DMPANL Raw Memory Selection

Figure 6–4 shows the syntax for DMPANL raw memory selection.

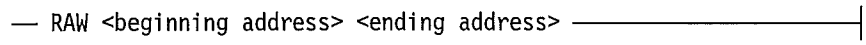


Figure 6–4. DMPANL Raw Memory Selection Syntax

RAW

This option prints memory from the specified beginning address to the specified ending address. The option does not print the system tables.

DMPANL Save File Assignment

To copy the MCP memory dump file as a permanent, multi-area diskpack file, use the syntax shown in Figure 6–5:

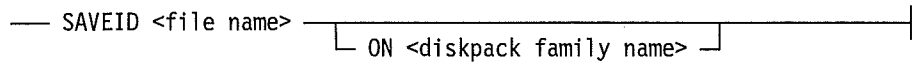


Figure 6–5. DMPANL Save File Assignment

If you do not specify a diskpack family name, the default is system resource pack.

DMPANL Task Selection

Figure 6–6 shows the syntax for DMPANL task selection.

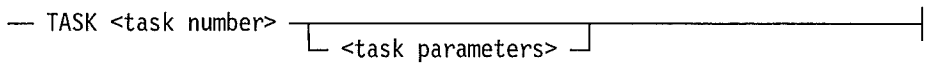


Figure 6–6. DMPANL Task Selection Syntax

TASK

This option prints information about the task, which is indicated by the number that follows the command. The default consists of all of the task memory areas and MCP data relating to the task.

Task Parameters

If you select TASK, you can specify the task parameters that appear in Figure 6–7.

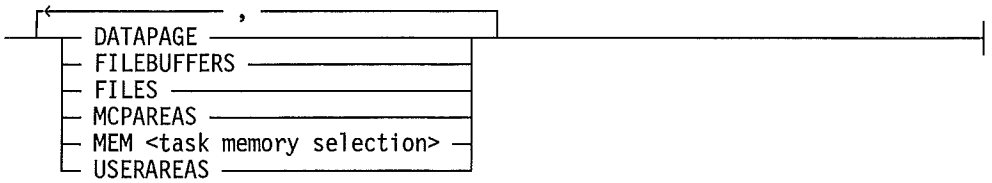


Figure 6–7. DMPANL Task Selection Parameters

DATAPAGE

This parameter prints the MCP data page for a task.

FILEBUFFERS

This parameter prints the same information as the FILES parameter and the external file buffers of any files that have them.

FILES

This parameter prints the information about all the files that are currently open or that were opened and closed without release.

MCPAREAS

This parameter prints just the MCP data related to a task.

MEM

This parameter prints a raw dump of the specified memory area for the task. To get a raw memory dump of a memory area if you did not specify MAT, the program uses the first MAT. Also, if you did not specify a memory area, the program uses memory area zero, which is the user data area. For code files generated by pre-V Series compilers, the only allowable MAT is the default. The beginning and ending addresses are specified in KD. If you do not specify them, the program prints the entire memory area.

USERAREAS

This parameter prints just the user data and code.

Task Memory Selection

Figure 6–8 shows the syntax for DMPANL task memory selection.

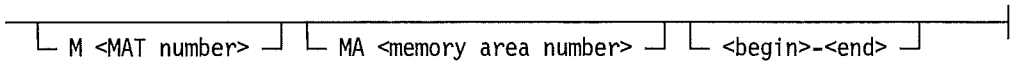


Figure 6–8. DMPANL Task Memory Selection Syntax

M <MAT number>

This option specifies the number of the desired Memory Area Table (MAT). The system defaults to the first MAT number if you do not specify a MAT. For code files generated before MCP Mark 1.0, the only allowable MAT is the default.

MA <memory area number>

This option specifies the number of the desired Memory Area (MA). If you do not specify a Memory Area number, the default will be memory area zero, which is the user memory area.

<begin>–<end>

This option specifies the beginning and ending addresses, in KD. The addresses are joined with a hyphen. If you omit this, the system prints the entire memory area.

Refer to the commands DM, DP, IL, and PM in Volume 2 for more information.

Section 7
DMPCPY—Memory Dump File Copy Utility Program

Overview

The MCP/VS operating system creates a memory dump file when you enter a DM command with a value of 0 or when the system fails. You can store the memory dump file on disk, disk pack, or magnetic tape. Because it is so large, many companies choose to store the file on tape. For information about how to direct a memory dump file to tape, refer to the USE record with the DUMP option in Volume 1 or the PM command in Volume 2.

After you store the memory dump file on tape, normal tape-handling utilities (for example SYSTEM/COPY or DMPALL) cannot copy the file onto other tapes because the format of a memory dump file is not compatible with that of the normal tape utility programs. To copy a memory dump file from one tape to another, use the DMPCPY utility program.

Executing DMPCPY

Perform the following steps to execute DMPCPY:

1. Mount the tape containing the memory dump file onto a tape drive. Use the RY command to ready the drive.
2. Enter the following system command:

```
EX DMPCPY
```

You do not need any special syntax to execute the DMPCPY program. It automatically searches for an input tape file named $00001. If a file with that name does not exist on tape, the system marks the program as waiting and waits until a tape with a file named $00001 is available.

When the program finds the input file, it requests a scratch tape to use for the output file. If the program encounters the end of the output tape before the end of the input tape a message appears on the ODT asking you to mount another tape.

When the program encounters an I/O error, the following message appears on the ODT:

```
ERROR ON INPUT TAPE. <MIX> AX TO CONTINUE
```

In almost all situations, having an exact copy of the memory dump file is important. For this reason, you should execute the DMPCPY program again with another tape or a different drive instead of continuing with errors.

Section 8
DMPMEM—Memory Dump Utility Program

Overview

Typically, the Fault Handler performs dumps for normal failures. In an abnormal failure, you can dump the MCP to tape, using the DMPMEM program, which is the memory dump utility that is floppy-disk based. DMPMEM is a control state program executed from the maintenance processor.

From a V 300 system, use the DMPMEM program to dump the MCP to tape. From a V 400 or V 500 system, load the DMPMEM program from the hard disk of the maintenance processor. When the DMPMEM program has dumped the MCP to tape, you can use the DMPANL program to analyze the MCP dump.

When the program writes to the tape, it writes first a header block and then a data block. The dump starts at the memory address 0 and continues to the end of MCP memory. The DMPANL program uses this header block to detect errors. This header block describes the data block, which is an exact duplication of data contained in a segment of memory. The tape that DMPMEM writes is always uncompressed, regardless of the setting of the system or unit COMPRESS option.

Executing DMPMEM

V 300 System

From a V 300 system, execute the DMPMEM program as follows:

1. Press the ON LINE/CONSOLE switch on the system cabinet so that the label CONSOLE on the switch lights up.
2. Insert the floppy disk named CV33nA (where n represents the release number) into one of the floppy disk drives in the system cabinet and enter the following command:

```
LOAD SYS DMPMEM; RUN
```

When the loading of the DMPMEM program is complete and execution begins, press the ON LINE/CONSOLE switch again so that ON LINE lights up. The system displays the following prompts:

```
Please enter tape drive channel & unit (cc/uu):
Please mount tape on cc/uu.
```

```
Tape <serial number> mounted.
O.K. to continue? Y/N
```

If the DMPMEM program cannot write to the tape, it tries again and displays the following message:

```
Tape write retry.
```

If this message appears frequently, use another tape. If an unrecoverable error occurs, the DMPMEM program terminates instead of producing an unreliable tape.

When the DMPMEM program accepts the channel/unit and the tape, the ODT displays the following message:

```
Memory Dump to tape Started
```

Refer to Section 6, "DMPANL—MCP Memory Dump Analysis Program," for more information about analyzing MCP dumps.

V 400 or V 500 System

From a V 400 or V 500 system, execute the DMPMEM program as follows:

1. Enter the following command at the maintenance processor:

   ```
   LOAD DMPMEM
   ```

2. Press the GO key. On a V 500 system, then press function key F8.

When the loading of the DMPMEM program is complete and execution begins, the system displays the following prompts:

```
Please enter tape drive channel & unit (cc/uu):
Please mount tape on cc/uu.
Tape <serial number> mounted.
O.K. to continue? Y/N
```

If the DMPMEM program cannot write to the tape, it tries again and displays the following message:

```
Tape write retry.
```

If this message appears frequently, use another tape. If an unrecoverable error occurs, the DMPMEM program terminates instead of producing an unreliable tape.

When the DMPMEM program accepts the channel/unit and the tape, it displays the following message:

```
Memory Dump to tape Started
```

When the DMPMEM program execution is complete, the following message appears on the ODT:

```
Memory dump to tape completed
Dismount and label tape on cc/uu.
```

After the program performs the dump, it terminates on a halt-branch instruction (OP=29).

Refer to Section 6, "DMPANL—MCP Memory Dump Analysis Program," for more information on analyzing MCP dumps.

Section 9
ECMANL—Environmental Control Module Analysis Utility Program

Overview

The MCP automatically invokes the Environmental Control Module Analysis utility program (ECMANL) when the ECM log file transfer occurs on a V 500 system. The ECMANL program formats the environmental information and creates a printer backup file.

Obtaining a Hard Copy of ECM Information

You can use the LN (Transfer and Print Log) command to print information about the internal operating environment of the V 500.

- Use the LNA command to analyze all logs, including the ECM Log.
- Use the LNE command to analyze only the ECM Log.

When you use the LNA or LNE command, MPCOPY automatically copies the log files to the V 500 disk.

Refer to Volume 2 for more information about the LN command. Refer to the *V 500 Maintenance Reference Manual* for more information about Environmental Control programs.

Section 10
LDCNTL—Pseudo Reader Load Control Utility Program

Overview

The Pseudo Reader Load Control Program (LDCNTL) enables you to create a disk or disk pack file, called a pseudo card file, from a card deck. A pseudo card file is also referred to as a control deck and can contain either data, MCP commands, or MCP commands and data. Although the LDCNTL program is a utility bound to the MCP, you can code and use your own version of the utility if you wish.

A pseudo card file created from a card deck can be useful for the following functions:

- Compiling programs that are punched on cards
- Running programs or program systems by punching the necessary commands on cards, and then reading the cards
- Carrying out commands that are punched on cards
- Reading data that is punched on cards
- Improving the efficiency of a card-based system

When the deck has been input and you have created the pseudo card file, you must activate the file before you can run it. The activation process depends on the procedure used to create a pseudo card file. If you use the manual procedure, use the RN or RNP command to activate the pseudo card file. If, on the other hand, you include the USE record with the APCR option, the MCP automatically activates the pseudo card file.

If you use cards, the pseudo card files improve the efficiency of your system in the following ways:

- Several card files can be read simultaneously, even though you have only one or two card readers.
- Data in a pseudo card file can be read faster than data read from a card reader.
- Files can be made accessible to many jobs simultaneously because the LDCNTL program can transfer your files to disk or disk pack devices.

If you use the pseudo card file more than once, follow the manual procedure to create your pseudo card file. Otherwise, the pseudo card file is purged from disk or disk pack after it is used. Refer to "Operating Procedures" in this section for more information.

LDCNTL System Commands and Options

When you use the LDCNTL program, you can use a number of system commands and MCP options. There are two types of system commands: MCP control instructions and keyboard input messages. You use these system commands to direct the MCP to perform particular actions.

You can enter MCP control instructions either on cards or from the keyboard, but you can enter keyboard input messages only from the keyboard. Volume 2 provides more detailed information about each of these command types, including the specific commands.

The following lists include the MCP control instructions and system commands used with the LDCNTL program.

LDCNTL MCP Control Commands

The following are MCP control commands:

- AX
- CHANGE
- CHARGE
- COMPILE
- DATA/DATAB (DATA CTLDCK)
- END/ENDCTL
- EXECUTE (LDCNTL)

LDCNTL Keyboard Input Commands

The following are keyboard input commands:

- AX
- CD, CDP
- CV, CVP
- DA, DAP
- DQ
- GO
- LD, LDP
- OL
- RD, RDP
- RN, RNP
- RO
- SD

- SO
- TO

Note: *The AX command is both an MCP control command and a keyboard input command.*

The MCP provides a number of options. You can specify a USE record to designate that a certain option is to be used. If you omit the USE record, the option is not used. Use the SO, RO, TO commands to set, reset, or interrogate an option.

Volume 1 provides more detailed information about the following USE records that affect the treatment of pseudo card files:

- APCR
- AURD
- PCRM

The LIMIT records set the sizes of certain MCP tables and the values of certain system parameters. Refer to Volume 1 for information about the limit on pseudo card files.

Determining When to Use Pseudo Card Files

If any of the following conditions apply, you should use card readers instead of pseudo card files:

- You only use card decks occasionally.
- You do not do a lot of multiprogramming work that involves cards. Multiprogramming means that you have more than one program running at the same time.
- Only a few of your programs use card files.

Other considerations include the following:

- The amount of time it takes to load a card file to disk or disk pack
- The amount of space a large card deck requires on the disk or disk pack
- The small amount of memory needed for the pseudo card file processing
- The requirement for punched-card equipment

Creating a Pseudo Card File

Use **one** of the following procedures to create a pseudo card file:

- Enter an LD or LDP command or an EXECUTE LDCNTL command.
- Enter CHANGE <file name> TO #. This command assigns a pseudo card file number to a disk or disk pack file; however, the file must already be in pseudo card file format. The MCP then assigns a number to the pseudo card file as described below. Write this number down so you can activate the pseudo card file later with the command RN or RNP. The following message then appears on your terminal:

  ```
  <file name> CHANGED TO #pnnnn
  ```

- Use the CANDE (Command and Edit) system to enter a COPY TO CTLD command. Refer to the *V Series CANDE Installation and Operations Guide* for more information.
- Write a user program that creates files in pseudo card file format. Refer to "Files and Records" later in this section for format requirements.

When you create a pseudo card file, the output disk or disk pack file identifier must be *#00000* even if you create a pseudo card file with a user program independently of the LDCNTL program. The special identifier #00000 tells the MCP to assign the next sequential pseudo card file number when a file is opened.

All pseudo card files have a name of the form *#pnnnn*.

- The pound sign (#) indicates a pseudo card file.
- The letter p represents the number of the system on which the file was created.
- The letters nnnn represent the number that the MCP assigns the file.

Requirements for a Control Deck

To read in a card deck from a physical card reader, you must first include two special cards that indicate where the deck begins and ends.

The first card in your card deck must be ?DATA CTLDCK. The last card tells the MCP that you want to end the LDCNTL program. Therefore, the last card in your last card deck must be ?ENDCTL.

The MCP only processes ODT control instructions as it creates the pseudo card file. ODT control instructions enable you to enter keyboard input messages from a card reader. The cards must be punched in the following format:

```
?SPO <text>
```

where <text> is the keyboard input message.

The MCP reads all other control instructions, including the ?END cards, as data. It does not execute these commands unless the pseudo card file is either automatically activated or manually activated and executed.

In the CTLDCK file, the deck includes all physical cards from the ?DATA CTLDCK card through the ?ENDCTL card. The data file includes all cards between ?DATA CTLDCK and ?ENDCTL (except ODT control instructions), even though some contain invalid characters in column 1.

If you include the USE record with the AURD option, pseudo card files with erroneous combinations of control instructions are removed automatically. If you do not include the USE record with the AURD option, the MCP displays the message CONTROL CARD ERROR followed by an ACCEPT message. You must then fix the error before you can use an AX command to tell the LDCNTL program to continue.

An example of an erroneous combination of control instructions is a ?END card that is not followed by a control instruction or the special ?ENDCTL card.

Initiating Load Control

You can initiate the LDCNTL program at any time, as described in "Creating A Pseudo Card File" in this section. The LDP command puts the pseudo card file on disk pack. The LD command puts the pseudo card file on disk, one file per card reader.

If you execute the LDCNTL program when you first initialize the system, you can leave it in the mix until processing is complete for the day. In this case, you would not use the ?ENDCTL card until you had read in the last card deck for the day. If the LDCNTL program does remain in the mix all day, it will be idle most of the time. However, the LDCNTL program does not consume much memory.

You can enter more than one LD or LDP command at one time. However, if you input more LDCNTL program messages than there are card readers, you use more memory without increasing productivity.

The MCP does not require that the LDCNTL program be in the mix to process pseudo card files.

The cards that you read in can be in EBCDIC or BCL format, or in both. However, you must put a ?DATA control instruction in front of those cards that are in EBCDIC format, or a ?DATAB control instruction in front of those cards in BCL format. The LDCNTL program cannot process BINARY decks.

If the input from which a pseudo card file is made comes from a Remote Job Entry (RJE) terminal, the program should execute a PROGRAM BCT to obtain its RJE link. Enter the RJE link value in columns 75–76 of the control instructions in the pseudo card file, a move of 2 UA to 2 UA, and put @FF@ in column 74. This link value provides a route back to the RJE station for messages that the pseudo card file generates.

MCP Capabilities in LDCNTL

A pseudo card file can contain as many as 100,000 card images. When you activate a pseudo card file, the MCP creates a record in the disk-resident Pseudo-Reader Cross Reference Directory (PCRXRF). The PCRXRF is capable of handling 80 active pseudo card files for each system. A total of 9999 pseudo card files can be on disk for each system; however, only 80 can be active simultaneously.

Use the SD command to further restrict the number of pseudo card files the APCR facility activates simultaneously. Enter *SD <integer>* where the integer is any value from 1 to 80.

If the directory becomes full or the operating system reaches the limit, the MCP temporarily stops activating pseudo card files. You can override this pause with the RN or RNP command, which activates more pseudo card files up to a maximum of 80 for each system.

Operating Procedures

If you want to use a pseudo card file without reading it in a second time, you must use the manual procedure. If you intend to use a pseudo card file only once, you can use the automatic procedure, which will then remove the pseudo card file after processing is complete.

The following steps outline the procedures for creating pseudo card files. Steps 1–5 apply to both the manual and the automatic procedures.

1. Punch the cards *?DATA CTLDCK* and *?ENDCTL.*
2. Place the ?DATA CTLDCK card at the beginning of the physical card file.
3. Place the ?ENDCTL card at the end of the deck unless you plan to use the LDCNTL program again later in the day. Refer to "Creating a Pseudo Card File" and "Typical Pseudo Card Decks" in this section.
4. Place the card file in the card reader and start it.
5. Enter *TO APCR* to determine if the USE record with the APCR option is included in the system configuration file.
   - If APCR = 1, the MCP will automatically create and run the pseudo card file.
   - If APCR = 0, you must use the manual procedure.
   - If you need to use the pseudo card file more than once, you must use the manual procedure.

Manual Procedure

1. Enter *RO APCR* to change the value from 1 to 0, if applicable.
2. Enter *LD* or *LDP* and transmit.
3. Enter *DQ* to obtain the deck number. You will need the deck number later.

   Note: *With the deck number, you can obtain the status of the pseudo card file by entering the CD, CDP, or OL commands. Use CD or CDP for inactive pseudo card files and OL for active pseudo card files.*

   - Enter *RN* or *RNP <deck number>* SAVE to activate and process a pseudo card file on the system from which you enter this command.
   - Enter *RN* or *RNP <system number> <deck number>* SAVE to activate a pseudo card file on another system.
   - Enter *RN* or *RNP = SAVE* to activate and process all pseudo card decks on the system from which you enter this command.
   - Enter *RN* or *RNP <system number>* = to activate and process all pseudo card files on another system.
   - Enter *RN* or *RNP A* = to activate and process all pseudo card files on all systems.

Note: *An active pseudo card file is accessible to all programs on the system on which it was processed.*

Automatic Procedure

1. Enter *LD* or *LDP* and transmit.
2. Enter *DQ* to obtain the deck number; you may need the deck number later.

   Note: *With the deck number, use the OL command to obtain the status of the pseudo card files that the MCP has activated.*

3. Run the appropriate programs to use the pseudo card file (the MCP automatically activates and processes the pseudo card file).
4. Put the ?ENDCTL card through the card reader when you are finished with the LDCNTL program.

Deleting Pseudo Card Files

You can use any of the following methods to remove a pseudo card file from the system:

- Use an RD or RDP command to remove inactive pseudo card files created manually. Before attempting to remove a pseudo card file, use the DA or DAP command to make it inactive.
- Include the USE record with the APCR option to have each pseudo card file automatically removed after it is processed. If the program reading the pseudo card file is discontinued or fails, the pseudo card file remains on the system.

- If a pseudo card file contains a source file for a COMPILE and GO operation, and if the compile fails or the system finds an ?END or ?DATA card during flushing of control instructions, the system removes the pseudo card file.
- Include the USE record with the AURD option to have pseudo card files that contain erroneous combinations of control instructions automatically removed from the system.

Recovery of Pseudo Card Files

When the system fails in a manner that requires a halt/load while pseudo card files are active, recovery of the active pseudo card files is automatic. The system searches the Pseudo-Reader Cross Reference Directory and reinitializes all active pseudo card files. However, it puts the pseudo card files in a not-ready state. Use the command RN or RNP to reactivate these pseudo card files regardless of the value of the USE record with the APCR option.

Typical Pseudo Card Files

Figure 10–1 shows a typical CTLDCK file. This example CTLDCK file contains only one data file and one set of control instructions.

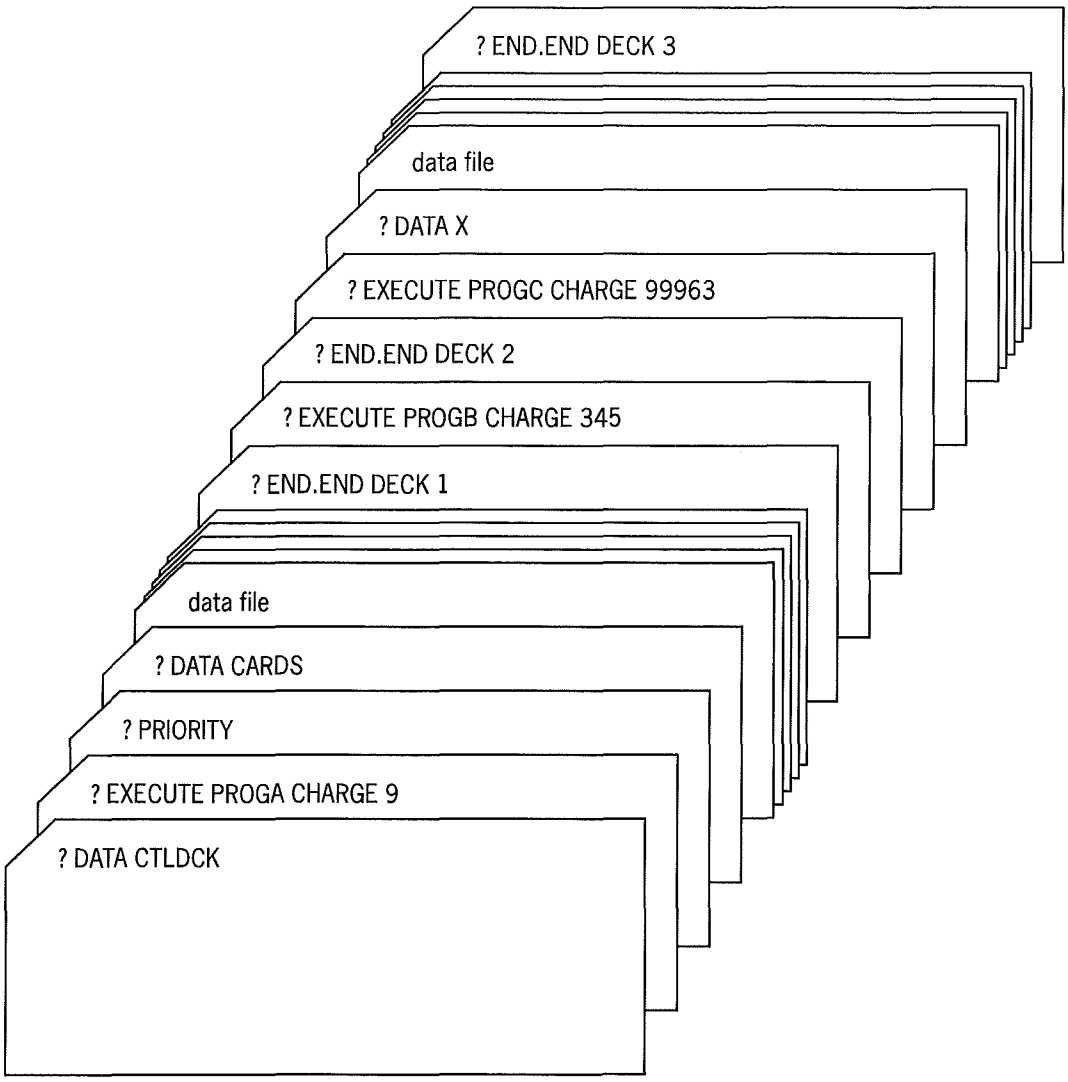


Figure 10–1. Typical Pseudo Card File

Note the first card, ?DATA CTLDCK. All the cards to be translated into the pseudo card file follow this card.

Note the last card, ?END.END DECK 3. This card indicates that this part of the LDCNTL program is finished, but that more card decks can be processed.

These cards are processed sequentially, in the exact order in which they are entered.

Note: *If you enter the LD or LDP command at the beginning of the day and plan to use the LDCNTL program all day, do not use the ?ENDCTL card until you pass the last card deck through the reader.*

You can create more than one pseudo card file out of a single card deck. Each time the MCP reads an ?END card, the pseudo card file is closed. If more cards follow the ?END card, a new file is opened, the MCP assigns the next number in sequence to the new file, and those cards are stored in that file.

A single pseudo card file can contain one or more data files or sets of control instructions. A pseudo card file that contains more than a single set of control instructions or a single data file is called a stacked pseudo card file. Stacked pseudo card files with more than one set of control instructions have a value in the link field. When it creates a stacked pseudo card file, the LDCNTL program enters this value, which is the record number of the next control card in the file.

Figure 10–2 shows an example of a stacked pseudo card file. Because the MCP processes the pseudo card file serially, the MCP does not process the second EXECUTE request until it reads all cards in the first data file.

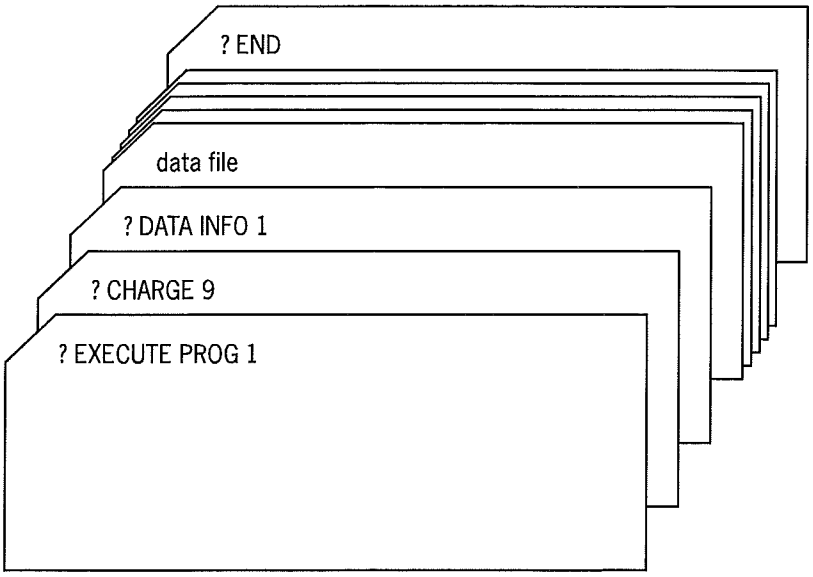


Figure 10–2. Execute Program

Files and Records

A pseudo card is a file consisting of card images with both program data and MCP control instructions. Table 10–1 shows the format of a pseudo card file.

Table 10–1. Pseudo Card File Format

File Attribute	Disk	Disk Pack
Record size	100 UA	120 UA
Blocking Factor	8	6
Records per Area	1000	1200
Number of Areas	100	100

A file must contain an integral number of blocks per area. The LDCNTL program creates pseudo card files on disk or disk pack. Table 10–2 shows the format of each record in a pseudo card file.

Table 10–2. Pseudo Card Record Format

Field	Size	Notes
Record Type	1 UN	D for control instructions; 8 for data cards
Flag Digit	1 UN	F (control instructions) specifies link present
Link	6 UN	Zero relative key to next control instruction
Data	96 UA	Card Image
Filler	20 UA	Present for disk pack file only

The record-type field contains the first digit of the result descriptor that occurred when the system read the card. D (undigit) means an invalid character appears on the card. The system replaces invalid characters in control instructions with an EBCDIC question mark character (6F).

The flag digit is present in control instructions when the control instruction link is present. The MCP automatically enters the link field value when it creates a stacked pseudo card file.

You can recognize control instructions by examining the result descriptor for the record. The first three digits of the result descriptor for control instructions are D40. The system uses an 800 descriptor for all other cards.

File Assignment

When the MCP processes an active pseudo card file, it processes all control instructions until it reaches a ?DATA statement. A ?DATA statement identifies the name of the data file. When a program needs the data file with that name, the MCP checks both the IOAT and the Pseudo-Reader Directory for the name of the data file.

Make sure that each data file has a unique name. The MCP assigns files as described in the following paragraphs.

File assignment for a card reader involves connecting of the program file information block (FIB) of the card file to the appropriate MCP IOAT entry and initially loading the program buffers.

Pseudo card file assignment requires the following activities. First, the MCP must obtain a 4 KD main memory buffer for pseudo card file use. In this memory area, the MCP builds an IOAT entry and DFH (Disk File Header) skeleton for the file. Internally, the MCP treats the pseudo card file as a type of disk file. The MCP marks the pseudo card file "in use" in the directory and associates it with the program FIB. Because the external memory area functions as buffers, the MCP modifies the program FIB to reduce the file to a single internal buffer. The MCP reads a block of card images into the external buffer; the MCP fills the program internal buffer, if necessary. This process completes the OPEN request.

Pseudo card file assignment requires the following activities:

1. The MCP must obtain a 4 KD main memory buffer for pseudo card file use.
2. In this memory area, the MCP builds an IOAT entry and disk file header (DFH) skeleton for the file. Internally, the MCP treats the pseudo card file as a type of disk file.
3. The MCP marks the pseudo card file "in use" in the directory and associates it with the FIB program.
4. Because the external memory area functions as buffers, the MCP modifies the FIB program to reduce the file to a single internal buffer.
5. The MCP reads a block of card images into the external buffer; the MCP fills the program internal buffer, if necessary. This process completes the OPEN request.

   READ operations cause the transfer of records from the external buffer to the program buffer and, if necessary, to the program work area. This transfer may be necessary because the MCP empties the external buffer when it initiates a disk read to refill it. With program execution, the system overlaps all I/O operations.

Deallocation

Deallocation means that the system terminates the association between a card reader or pseudo card file and the program using the data file, not that the card reader or pseudo card file is deactivated or removed. A pseudo card file is deallocated when the system detects an end-of-file (EOF) condition, when the program executes a CLOSE request before the EOF, or when the program terminates before either event occurs.

EOF detection means that a control instruction (other than an ODT control instruction) appears while the program is associated with the device. Unlike other devices, card readers are detached from the program at the point of EOF rather than at a CLOSE request. For pseudo card files, the MCP releases all memory space allocated to the pseudo card file and performs certain maintenance functions on the Pseudo-Reader Directory entry.

If the program executes a CLOSE request or terminates normally before it reaches EOF, the MCP deallocates the card reader or pseudo card file and flushes it to the next control instruction. The system flushes a card reader by passing cards through the card reader until it reaches the next control instruction or until it makes the card reader "not ready."

Pseudo card file flushing achieves similar results, but the system accomplishes it in a different manner. When the pseudo card file is ready for flushing, the MCP examines the next card image in the pseudo card file.

- If the card is a control instruction, the MCP stops the flushing operation.
- If that card is not a control instruction, the MCP uses the control instruction link found in the last DATA statement to flush the pseudo card file. This link provides a rapid method of finding the next control instruction without the overhead of reading each record from the disk.
- If the link is not present, the MCP attempts to find the next control card by manually reading each card.
- If the program terminates abnormally (for example, with a DS command), the MCP deactivates the pseudo card file and removes it from the Pseudo-Reader Directory, but does not remove it from the Disk Directory. This puts the pseudo card file in a not ready condition and you must use the command RN or RNP to reactivate it.

LDCNTL—Punch Backup to Pseudo Card File Conversion

Use the command CV or CVP to initiate LDCNTL to convert a file in punch backup format to a file in pseudo card file format. Doing so is useful when existing programs generate card files that will be read by subsequent programs, but you do not need the cards.

The system always converts punch backup files to pseudo card files on the same media, disk or disk pack. To keep the punch backup file in addition to the pseudo card file, specify SAVE. If you specified CHRG ALL at the time of cold-start, you must also include the CHARGE parameter instruction. The system does not provide special handling of card punch records selected by the stacker.

Enter one of the following:

```
CV <deck number> SAVE
```

or

```
CVP <deck number> SAVE
```

or

```
CV <deck number>
```

or

```
CVP <deck number>
```

The MCP automatically enters the following parameters into the LDCNTL program:

BASE. + 32	5 UN	CV deck number
	1 UN	flags
	8 bit	CV command
	4 bit	pack media
	2 bit	<unused>
	1 bit	save flag

If you create a user program called LDCNTL, the MCP also enters these parameters into your program automatically.

Section 11
LDHOST—Host Load Intrinsic Program

Overview

The Host Load Intrinsic Program (LDHOST) downloads firmware to a programmable controller, a data communications processor (DCP), or a Uniline device. For a B 974 DCP, the LH command invokes the utility program B 974LD. The system does not permit you to download firmware to the primary MCP disk channel.

After the load is completed, the LDHOST program displays the release level of the firmware that was just loaded if you loaded a disk pack/disk channel.

If the LDHOST program fails to load any device, the program leaves the channel of that device inhibited and notifies you that the load failed and that the channel is inhibited. The LDHOST program also displays the segment of device memory that was being loaded at the time of the failure, the result descriptor (R/D), and the number of digits actually transferred.

You need not load SMD DLP II. The SMD stands for Storage Module Device, and DLP stands for Data Link Processor.

Host Load for B 974 DCPs

The *B 974 DCP Software Installation and Operations Guide* provides procedures for loading the B 974. In some cases, you use the LH command to initiate the B 974LD utility program. In other cases, you use the SO command with the DCP option, which does an automatic LH. For more information about the LH command, refer to Volume 2. For more information about the SO command with the DCP option, refer to Section 27, "Data Communications Processor (DCP)."

Host Load for B 874 DCPs

When you invoke the Host Load program to load a B 874 DCP, the system automatically dumps the S-memory of the DCP. This dump appears as a listing called S-DUMP and a disk file called DCxxpP: where *xx* represents the channel of the B 874 DCP and *p* (lowercase) represents the system number of the host.

Automatic Operation

The MCP can schedule the LDHOST program automatically when a disk or disk pack controller fails. However, the following conditions are necessary before this can happen:

- You must have previously designated one of two conditions:
  - Entered LH <channel> <firmware name> to associate the name of the firmware file that is to be loaded to the controller and the number of the channel to which the controller is connected.
  - Specified the name of the firmware file on a DLP record in the system configuration file.
- The firmware file to be loaded to the controller must be on disk.
- The firmware file must be accessible (that is, it must not be on the disk where the controller failed).

If the MCP determines that it can load the controller, one of the following messages appears on the system console:

```
CHANNEL <channel number> FIRMWARE LOAD SCHED
```

In this case, you need not do anything.

```
CHANNEL <channel number> FIRMWARE LOAD REQD
```

In this case, you must determine the problem and take the appropriate corrective action.

The MCP automatically enters the following parameters into the LDHOST program:

BASE 32 UN=	Firmware ID	6 UA
	Primary Channel	2 UN
	Alternate Channel	2 UN
	B 874 DCP S-Memory Size (HEX)	4 UA
	S-Memory Dump Flag	1 UN
	Inhibit Channel Flag	1 UN
	Hardware Type	2 UN
	Unit	1 UN
	DCP "WARM" Load Flag	1 UN
	Hardware Subtype	1 UN
	Filler	1 U

Host Load for DCDLP

When you invoke the Host Load program to load a DCDLP, the system might automatically dump the memory of the DCDLP. The memory dump occurs in the following conditions:

- Memory is always dumped when you use the LH command to load the DCDLP.
- If you use the SO DCP command to load the DCDLP, the Host Load program reads the status word of the DCDLP. If the status word indicates that the DCDLP has been cleared, then the Host Load program dumps the memory of the DCDLP.

If an I/O error occurs during the memory dump, the Host Load program terminates the memory dump and proceeds with the loading of the DCDLP.

Loading the LOS into an Intelligent Laser Printer

An Image Page Printer (IPP) is one type of intelligent laser printer. Perform the following steps to load the Loadable Operating System (LOS) into an IPP.

1. Make sure that the device is online and is in the Resident Operating System (ROS) mode.
2. Enter *LH <channel number>* to load the LOS.

Refer to the *B 9290–30 Intelligent Laser Printing System Operator's Guide* for operating instructions.

Section 12
LOADFW—Offline Firmware Loader Program

Overview

Use the Offline Firmware Loader Program (LOADFW) before you perform a cold-start to load firmware to disk pack controllers, shared system processors (SSPs), and Uniline DLPs. The LOADFW program performs the initialization, verification, and relocation (IVR) functions for LAK disk packs on an individual basis. With a V 300 system, this program resides on a floppy disk. With a V 400 or V 500 system, this program resides on the hard disk of your maintenance processor.

The LOADFW program loads the firmware files from a tape or a disk pack. If the firmware is on a tape, it must be a SYSTEM/COPY tape with the default buffersize of 18000 digits. If the firmware is on disk pack, the disk pack must have a standard label. If the firmware is on a version 2 disk pack family, it must be on the base pack of that family.

The LOADFW program loads firmware to the following types of devices:

- Uniline DLPs
- Shared system processors (SSPs)
- Disk or disk pack controllers–BX383, BX384, BX385, B9387

The LOADFW program can also perform the IVR functions for LAK disk packs that are shipped in 180-byte format. This capability is useful for systems that need a disk pack in 100-byte format to use as MCP disk. IVR performs the following functions:

- **Initialization.** Initializes the track and sector structures on the disk pack into a 100-byte format.
- **Verification.** Verifies that the track and sector structures of the disk pack have been initialized properly.
- **Relocation.** Relocates and thereby removes bad sectors from use.

Also, you can use the utility program DISPKV to perform IVR functions on disk packs in LAK mode. DISPKV can be more convenient because it is used while the MCP is running. Refer to Section 3, "DISPKV–Disk Pack Utility Program," for more information. You cannot use DISPKV for MD4 SMD LAK; you must use PTDMNO instead.

You operate the LOADFW program from an ODT linked to the console DLP or an Operating Control System (OCS) linked to a Uniline DLP. First you press the ON LINE switch on the system cabinet to put in console mode; then you load the program from the

floppy disk or the maintenance processor into memory. When the program is in memory, you execute it, using various commands, depending on the functions required.

Execution of LOADFW

Use one of the following procedures to execute the LOADFW program, depending on the system you are using.

V 300 System

1. Insert the floppy disk named CV33nA (where n represents the release level) into the primary floppy drive.
2. Make sure the ODT is offline and in console mode by pressing the ON LINE/CONSOLE switch on the system cabinet so that the word CONSOLE on the switch lights up. The system should be stopped (as indicated by the ODT status heading). If it is still running, press the SPCFY key or enter the TERM command to stop the system. Wait until the ODT status heading indicates that the system has stopped.
3. Enter the following command:

   ```
   LOAD SYS LOADFW; RUN
   ```

   The ODT set up with the SO command and the OCS option will be used for communication. For this reason, you must set up the command beforehand. Do not use a SO command with the OCS option for a Uniline OCS that does not have firmware loaded to it.
4. If you intend to use the ODT for communication, be sure that the ODT is online. Do so by pressing the ON LINE/CONSOLE switch on the system cabinet, which causes the word ON LINE to light up. The program then begins the dialog by prompting you for the channel and unit of the tape or disk pack that contains the firmware files.

V 400 and V 500 Systems

1. Enter the following command:

   ```
   LOAD PROGRAM
   File Name (Def Dir=[Sys] <Program>)___loadfw_____________
   [Load SYS? (Y/N, Def = N)]___Y_____________________________
   ```

2. Press the F8 function key. The system loads the LOADFW program into memory from the maintenance processor. The system changes from STOPPED to RUNNING status, and then the LOADFW program begins to execute.
3. Enter the channel and unit number of the tape or disk pack that has the firmware file.

   (channel number/unit number)

4. Enter the name of the firmware file to be loaded. The LOADFW program will find the firmware and load it into memory.

   (firmware file name)

5. When the LOADFW program notifies you that the firmware has been loaded into memory, enter the channel number of the device that the firmware is to be loaded to.

   (channel number)

6. When the LOADFW program displays a message that the firmware has been loaded into the device, perform one of the following:

   - To load the same firmware to another device of the same type, enter another channel number.
   - To load a different type of firmware to a different type of device, enter the name of another firmware file.
   - To use a different tape or disk pack firmware source, enter the channel/unit.
   - To load a disk controller, enter *INI* or *VER* and the unit number. INI both initializes and verifies the disk pack; VER only verifies it.
   - To quit the program, enter END.

While the program is running, various prompts appear on your terminal.

The LOADFW program displays error messages when it cannot run successfully. Refer to "Error Messages" later in this section for more information.

Normal Messages

The following messages, which are listed in alphabetical order, report on program status or ask you to enter information.

```
Canceling channel . . .
```

The LOADFW program is canceling the channel. This message is normally sent when SSP DLPs are loaded. There is a 10-second delay to allow the DLP to reinitialize.

```
<cc/uu> <LAK type> {initialized} with no error(s), no relocate(s)
                   {verified} with no error(s), no relocate(s)
```

The LAK was successfully initialized or verified.

```
<channel> SSP cleared and unlocked
```

The LOADFW program loaded the SSP successfully. The SSP is clear and initialized.

```
Enter channel to be loaded
```

This message prompts you to enter the number of the channel to load.

```
Enter firmware name
```

This message prompts you to enter the name of the firmware file to be loaded.

```
Enter input tape or disk pack cc/uu
```

This message prompts you to enter the channel and unit numbers of the tape drive or disk pack that has the firmware files.

```
Enter: <channel>.......          for next channel to load,
<name>......                     for new firmware,
<channel/unit>.......            for new firmware source,
or END

Enter:<channel>.........         for next channel to load,
<name>........                   for new firmware,
<channel/unit>........           for new firmware source,
INI <unit>........               to initialize,
VER <unit>........               to verify,
or END
```

The LOADFW program displays the two preceding prompts when it completes an operation. The program wants to know what you want it to do next. At this point you can instruct it to do one of five things, depending on which of the following you enter:

- To load the same firmware to another channel, enter the channel number. To load another firmware file, enter the firmware file name.
- To designate a new channel and unit number of the tape or disk pack that has the firmware file, enter the channel or unit number.
- To initialize an LAK, enter the INI command and its unit number.
- To verify an LAK, enter the VER command and its unit number.
- To quit the program, enter the following:

```
END  {Initialization}
{Verification}<percent> complete
Time remaining: <minutes> minutes, <seconds> seconds
```

During an LAK initialization or verification, you receive this progress report:

```
Reading firmware ...
```

The LOADFW program is searching the input medium for the requested firmware file.

```
Loading firmware ...
```

The LOADFW program is loading the requested firmware.

```
(<release code>) phase {1 }  {2 }  {3 } loaded  {all}
```

The LOADFW program has completed the requested firmware load operation.

```
EOJ LOADFW
```

The LOADFW program has gone to end-of-job.

Error Messages

The LOADFW program displays the following error messages.

- <cc/uu> invalid label
- <cc/uu> not ready
- <cc/uu> unknown input medium tape
- <cc/uu> does not have <firmware file name>
- <cc/uu> read error continue?
- Corrupted directory/header link

The LOADFW program displays one of these messages when it cannot load firmware. The actual message it displays indicates why it cannot perform the function.

```
<cc/uu> not LOADMP, PACKUP or SYSTEM/COPY tape
```

The device on channel/unit has the wrong tape format. The tape must have a LOADMP, PACKUP, or SYSTEM/COPY tape format. The SYSTEM/COPY format is recommended.

```
Enter input tape or disk pack <cc/uu> again
```

The program displays this message when it wants you to reenter the channel and unit number. Frequently, this message appears after another error message that refers to a previously used tape or disk pack.

```
<cc/uu> got invalid I/O on load
```

An I/O error occurred when the LOADFW program attempted to load the firmware.

```
<cc/uu> is not a LAK or initializable media
```

Either the device on that channel is not LAK disk pack in 100-byte format, or it is the wrong type of device. The LOADFW program cannot initialize or verify MD4 SMD disks. To initialize MD4 disks, use the peripheral test driver (PTD) program SMDIVR.

```
<cc/uu> <LAK type> {initialized}
                   {verified} with <number> errors, <number> relocates
```

After the program finishes an initialize or verify operation, it displays this summary.

```
DLP not an HT, SSP, or Uniline -- enter channel:
```

The LOADFW program determines the device type with a TEST ID operation. The program can load either a host transfer, SSP, or Uniline DLP. This message usually indicates that the requested channel is other than one of these three types. The message can appear when a DLP is hung. Refer to Section 4, "DLPXCO and DLPXNO–DLP Utility Programs," for information about clearing a hung DLP.

```
Error on sector <number> in cylinder <number> of <max cylinder number> Enter
(I)gnore, (R)elocate, (E)nd or blank for retry
```

During an initialize or verify operation, the LOADFW program found an error on the hardware. Enter one of the following choices:

I—Ignore and continue with the next cylinder.

R—Relocate the bad sector.

E—End to terminate the operation.

Blank to retry the operation.

```
Fatal IOP error occurred <cc/uu> IOP R/D = <R/D>
```

The LOADFW program found an IOP error. The result descriptor (R/D) involved appears on the second line of the message.

```
Segment <number> not loaded - <number> of digits passed
```

The LOADFW program stopped just before the indicated segment number. The number of digits passed indicates the amount of firmware loaded.

Section 13
MAKTRN—Translation File Generator Program

Overview

You can perform the following functions with the MAKTRN program:

- Create a complete new collating sequence for sorting
- Interchange specific characters, retaining the remainder of the normal collating sequence
- Assign the same collated value to multiple characters

The MAKTRN program creates a disk file used with the key translation option of the SORT intrinsic utility programs (SORT. and SORT:). In COBOL, this option is called the COLLATING SEQUENCE. The file that the MAKTRN program creates is a 400-byte, single-record, single-area file on 100-byte disk. The file contains a translate table in the format expected by the SORT intrinsic programs.

The MAKTRN program reads parameter records that describe either an entire collating sequence or only the non-standard portions of a collating sequence. The program checks the parameter records for correct syntax. If it does not find any syntax errors, the MAKTRN program creates a valid translate table.

Operating Instructions

Use the following pseudo deck or control cards to execute the MAKTRN program:

```
?EXECUTE MAKTRN
?DATA CARD <parameter records>
?END
```

MAKTRN Parameter Records

The format of <parameter records> is as follows:

- All parameter records have a dollar sign ($) in column 1.
- A valid MAKTRN option name must appear in columns 2 through 5.
- Columns 6–71 are free form. You cannot continue characters from one record to another record.
- The MAKTRN program does not use Columns 72 through 80.

MAKTRN Options

The valid MAKTRN options are IDNT, ALFA, NUMR, and SEQN.

IDNT Option

This option determines the file name of the translate file you create. IDNT is not required. If you omit this option, the MAKTRN program assigns a default name of TRANS*. The IDNT option must be the first option in the parameter records.

Example

```
$IDNT TRANS1
```

This example assigns the name TRANS1 to the translation file that the MAKTRN program creates.

ALFA Option

This option indicates changes in a collating sequence. To translate a character to a different character, include the old and new characters next to each other, without intervening spaces. At least one space must appear between pairs of characters. The ALFA option is a convenient method of converting graphic characters.

Characters not included on ALFA (or NUMR) option records retain their standard position in the collating sequence. A character can appear as the first part of a pair only once. A character can appear as the second part of a pair as many times as needed. This enables you to collate any number of characters to the same character.

Example

```
$ALFA ([ )] -/
```

This example collates parentheses as brackets and dashes as slashes.

NUMR Option

This option uses two-digit numeric values to represent the old and new characters. You can use the NUMR option for any characters, but it is most convenient for non-graphic characters. As with the ALFA option, enter old and new characters in contiguous pairs. Separate the pairs with spaces.

Characters not included on NUMR (or ALFA) option records retain their standard position in the collating sequence. A character can appear as the first part of a pair only once. A character can appear as the second part of a pair as many times as needed. This enables you to collate any number of characters to the same character.

Example

```
$NUMR 0040 81C1 82C2 83C3
```

This example collates NULL characters (00) as spaces (40) and lowercase letters a, b, and c as uppercase A, B, and C.

SEQN Option

This option directs the MAKTRN program to create an entire new collating sequence in the translate file. If the SEQN option appears in the parameter records, you cannot use the ALFA and NUMR options.

The SEQN option requires that all 256 possible character combinations be included on the record. After you enter the keyword $SEQN, enter the character that collates as 00, then enter the character for 01, and so forth. A hyphen between characters denotes a range of values. Be sure to separate ranges of characters or single characters with spaces.

Example

```
$SEQN 40-EF F0-FF 00-3F
```

This example creates a collating sequence of spaces, special characters, letters, numbers, and data communication control characters.

Section 14
MDCOPV—Floppy Disk Copy Utility Program

Overview

The MDCOPV utility program copies files to and from floppy disks and performs maintenance functions on floppy disks.

The information is stored on the floppy disks, using the industry-compatible minidisk (ICMD) format or micro-minidisk unit (MMDU) format and depending on whether you have a system that uses 5.25- or 8-inch floppy disks. MDCOPV automatically determines the type of system it is running on and selects the corresponding storage format, ICMD or MMDU. MDCOPV performs the following basic functions:

- Copy function-copies files or an image of an entire floppy disk to and from a floppy disk.
- Analysis function-initializes floppy disks and removes files. Reports the amount of space used by files and the amount still available.
- Conversion function-converts object code or A Series S-code files to MMDU compatible format.

Supported Hardware

MDCOPV functions under the release level of the MCP/VS operating system covered in this document. The only V Series systems that require MDCOPV are V 300 systems. MDCOPV can also function in other environments, subject to the following system and hardware requirements:

System	DLP	Floppy Disk Format	Storage Capacity	Required Unit Number
V300/B4900	UC-DLP	5 1/4 MMDU	(640KB)	2 or 3
B2900/3900	UC-DLP	5 1/4 ICMD-LAK	(246KB)	2 or 3
B2900/3900	UC-DLP	8 ICMD	(246KB)	2 or 3
Pre-900 sys	ICMD-DLP	8 ICMD	(246KB)	0 or 1

All copy functions require an initialized floppy disk. Refer to the IN option under "File and Floppy Disk Analyze Functions" later in this section. The ICMD-LAK format floppy disks cannot be initialized with the MDCOPV program. You must use the DIAG20 maintenance program, which runs in the universal console, to initialize these floppy disks.

Because of hardware limitations, the MDCOPV program can access only floppy disk drive units 0 and 1 when running through a control or ICMD-DLP, or units 2 and 3 when running through a UC-DLP. You must include the following records in the system configuration file before the MCP can access the floppy disk drives:

- The DLP CONSOLE record
- The UNIT NST record

The input and output channel numbers for these records is that of the console DLP for MMDU format floppy disks.

Valid unit numbers for floppy disks attached through controls or HT-DLP are "0" and "1". Valid unit numbers for floppy disks attached through a UC-DLP are "2" or "3".

Format Information

Industry-compatible minidisk (ICMD) format floppy disks must use the industry compatible format for floppy disks described in IBM publication GA21–9190–1. These floppy disks have a maximum record size of 128 bytes and a blocking factor of 1.

Industry-compatible look-alike (ICMD-LAK) format floppy disks are a combination of MMDU format and ICMD format. The floppy disk is initialized in MMDU format (256 bytes per sector) and used as if it were ICMD format (128 bytes per sector). Micro-Minidisk Unit (MMDU) format floppy disks have a maximum record size of 256 bytes and a blocking factor of 1.

Operating Instructions

To use the MDCOPV program, you must first execute the program with the EXECUTE command and then use the AX command to enter commands. The AX command associates commands input to the executed program with a mix number. When you execute the MDCOPV program, write down the mix number. Use the following syntax to execute the MDCOPV program:

```
EX MDCOPV
```

When the execution of the MDCOPV program is complete, the program displays the following message on the ODT:

```
ENTER INPUT PARAMETERS FOR MDCOPV
```

Enter the mix number and the commands you want in the following manner:

```
<MIX>AX"CH2 12 MINI/FINBAR HOMER BYPASS"
```

After you enter the commands and the system completes the processing, the MDCOPV program displays the same message again to indicate that it is ready for more input.

```
ENTER INPUT PARAMETERS FOR MDCOPV
```

You can input another command string, or you can input the following command to stop processing:

```
<MIX>AX END
```

The prompts and the messages that the MDCOPV program displays while it is processing, appear in this section.

Executing MDCOPV and Setting Switches

Program switch 1 (SW1) sets the length of the header record length field, which is used by the Purge and Remove functions.

SW1, with a value of 0, sets the header record length to 80 bytes, which is the default. Execute the MDCOPV program without setting any switches to get this record length.

SW1, with a value of 1, sets the header record length to 128 bytes for the ICMD format or 256 bytes for the MMDU format.

Example

```
EX MDCOPV; VA 1 100000
```

In this example, the MDCOPV program is executed and SW1 is set to a value of 1, which makes the header record length 128 bytes for ICMD or 256 bytes for MMDU.

Copy Functions

The following paragraphs discuss the copying functions that the MDCOPV program performs. These functions include copying files to and from a floppy disk and copying pseudo card reader files. Refer to "Copy Function for Floppy Disk Image Files" in this section for information about copying floppy disk image files.

Note: *In all cases, the MDCOPV copy function requires that the floppy disk be initialized before it can be used. Refer to the IN option under "File and Floppy Disk Analyze Functions" later in this section for more information about initializing floppy disks.*

Copying Files to a Floppy Disk

When you copy files to the floppy disk, the MDCOPV program checks for the largest available area on the floppy disk. If no space is available, a message appears on the ODT. Otherwise the MDCOPV program writes the file. If the file is too big, the MDCOPV program writes as much as it can and then requests that you insert another floppy disk so that it can continue to write the file.

For ICMD and ICMD-LAK format floppy disks, the maximum record size is 128 bytes. For MMDU format floppy disks, the maximum record size is 256 bytes. You must be careful when you copy files to a floppy disk with a record size greater than these maximums. The MDCOPV program displays a warning message stating that the record size is greater than the maximum and that, if allowed to continue, record truncation will occur. To continue, you must use the AX command to enter Y. If you want to stop processing, you must use the AX command to enter N.

Copying Files from a Floppy Disk

When you copy files from the floppy disk to disk or to disk pack, the copied files are created according to the options you entered when you executed the MDCOPV program. If you did not use output file specifications, the new file has a default record size, blocking factor, number of areas, and records per area.

The MDCOPV program can copy files into pseudo card reader (PCR) format. This feature enables you to create data files or execution decks on floppy disk. Then when these files are copied from the floppy disk, the MCP can activate them through standard pseudo card reader processing.

The MDCOPV program interrogates the file being copied for control records. If the first record is not a control record (?DATA or ?EX, for example), the MDCOPV program puts a ?DATA <file name> record at the front of the pseudo card file. The file name is that specified in the input string. The MDCOPV program also creates a ?END record at the end of the file if one is not already there.

Duplicating Floppy Disks

When the MDCOPV program duplicates a floppy disk, it makes an exact copy of the disk. The program starts at the first sector on the floppy disk and copies all the sectors thereafter. When the copy function is complete, the label, tables, and data are identical on the original and duplicated disks.

Command Syntax of the Copy Function

Figure 14–1 shows the command syntax for copying files to and from a floppy disk. The copy function syntax can copy files from 100-byte disk and 180-byte disk pack to the floppy disk. Conversely, it can copy files from a floppy disk to 100-byte disk and 180-byte disk pack media.

You can also use this syntax to copy pseudo card reader files and floppy disk image files to and from the floppy disk.

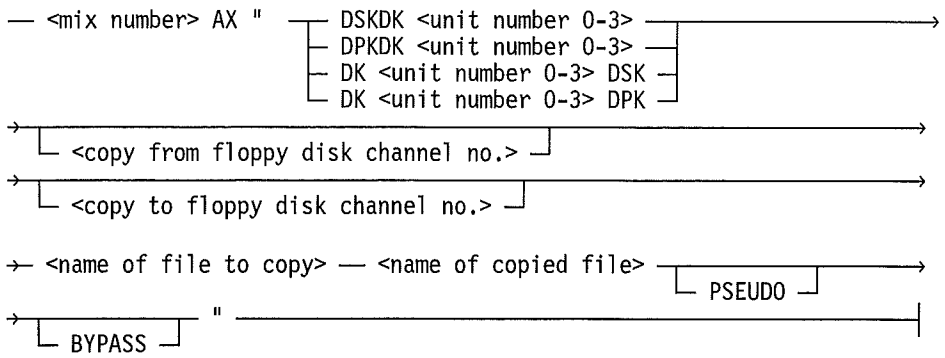


Figure 14–1. MDCOPV Floppy Disk Copy Syntax

<mix number> AX "

When you execute the MDCOPV program, write down the mix number. Enter the commands for the program with the mix number and the AX command. The commands for the MDCOPV program must be enclosed in quotation marks.

DSKDK<unit number 0–3>

Use this option to indicate that the file is to be copied from a 100-byte disk to the floppy disk drive that has the specified unit number.

DPKDK<unit number 0–3>

Use this option to indicate that the file is to be copied from a 180-byte disk pack to the floppy disk drive with the specified unit number.

DK<unit number 0–3>DSK

Use this option to indicate that the file is to be copied from the floppy disk drive with the specified unit number to 100-byte disk.

DK<unit number 0–3>DPK

Use this option to indicate that the file is to be copied from the floppy disk drive with the specified unit number to 180-byte disk pack. Copy the file to a disk pack family by using a multifile name.

<copy from floppy disk channel number>

Use this option to indicate the number of the channel to which the floppy disk drive unit is connected. This channel must be declared to the system with the hardware type NST (non-status device). The input and output channel numbers are those of the console DLP for MMDU format floppy disks.

<copy to floppy disk channel number>

Use this option to indicate the channel number to which the floppy disk unit is connected. The floppy disk channel must be declared on the system and have a hardware type of NST (non-status device).

The default output channel number is the input channel number specified.

<name of file to copy>

Use this option to specify the file name in this form:

<multifile name>/<file name>

The multifile name, which can be the name of a disk pack family, is optional. The file name can be from 1 to 8 characters in length for a file on the floppy disk. It can be from 1 to 6 characters in length for a file on disk or disk pack. You can specify a multifile name entry for disk, disk pack, or floppy disk files.

If you use the multifile name, it can be any name from 1 to 6 characters in length. When you use a multifile name, enclose the entire command string in quotation marks. The multifile name for floppy disks is the volume ID referred to in the volume label.

If you do not use the multifile name, the system uses the following default values:

- Floppy disk—all blanks (EBCDIC)
- DPK—system default disk pack

<name of copied file>

Use this option to indicate the name of the copied file, which takes the form <multifile name>/<file name>, where the multifile name is optional.

Refer to the option called *name of file to copy* for more information about multifile names.

PSEUDO

Use this option only to copy files to disk or disk pack. It indicates that the output file being copied is in pseudo card reader deck format. This option assigns the file a pseudo deck number that enables the MCP to activate the file in normal pseudo card reader processing.

BYPASS

Use this option only for files on floppy disk. When the bypass indicator is set, you cannot reset it.

The bypass option provides two capabilities:

- It enables you to set the bypass indicator in the File Header when you are copying a file to the floppy disk.
- It enables you to access a floppy disk file when the bypass indicator is set.

If you begin a command string with a quotation mark, you must end it with one.

Duplicating Floppy Disks with the Copy Function

Use the command syntax shown in Figure 14–2 to duplicate a floppy disk.

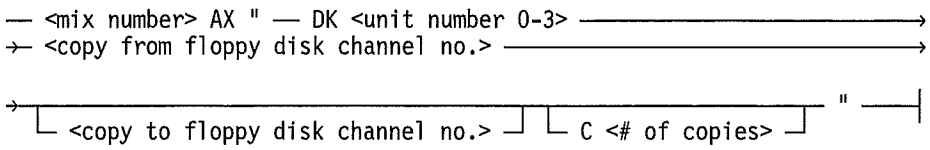


Figure 14–2. MDCOPV Floppy Disk Duplication Syntax

DK<unit number 0–3>DK<unit number 0–3>

Use this option to indicate that you plan to copy the entire floppy disk from the floppy disk drive with the specified unit number to another floppy disk drive with its specified unit number.

C<# of copies>

Use this option to indicate the number of times you want to duplicate a floppy disk. Enter *C* followed by the number of copies you want. For example, C5 requests five copies. The value can be up to three digits in length. The default value is one copy.

File and Floppy Disk Analyze Functions

Use the command syntax shown in Figure 14–3 to analyze the status of the floppy disk. This command syntax performs the following functions:

- Lists the files on the floppy disk
- Displays the volume ID of the floppy disk
- Prints a status listing of the floppy disk
- Changes a file name on the floppy disk
- Renames the volume ID of the floppy disk

- Purges the floppy disk
- Removes a file from the floppy disk
- Initializes a floppy disk

Before you enter these commands, you must execute the MDCOPV program. When you execute the program, write down the mix number.

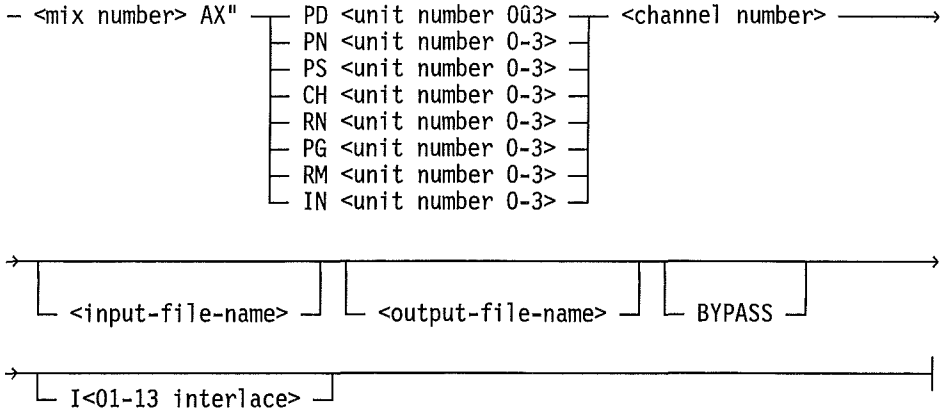


Figure 14–3. MDCOPV Flexible Disk Analyze Syntax

<mix number> AX "

When you execute the MDCOPV program, write down the mix number. Enter the commands for the program with the AX command. The commands for the MDCOPV program must be enclosed in quotation marks.

PD

Use this option to indicate the files that are on a floppy disk.

The names of all the files on the floppy disk appear on the system ODT. The floppy disk volume ID also appears. If the floppy disk does not contain any files, a message appears on the ODT.

PN

Use this option to indicate the volume ID of the floppy disk.

The volume ID of a floppy disk (multifile name) appears on the system ODT. The MDCOPV program requires the volume ID when it performs file maintenance or copy functions. The default value for volume ID is all blanks.

PS

Use this option to print a status listing of the floppy disk. This listing includes the volume ID, the file names, the beginning and ending of each file, and the total space available.

The MDCOPV program creates a printer backup file with the name MDSTAT. This file gives the exact status of a floppy disk. This listing contains a detailed analysis of the floppy disk directory along with pertinent volume label information. It also contains all the information about the files on the floppy disk; for example, BOE (beginning-of-extent), EOE (end-of-extent), bypass indicator, and EOD (end-of-data). The MDCOPV program also prints the total available space and largest available area on the floppy disk.

CH

Use this option to change the name of the file on the floppy disk.

RN

Use this option to rename the volume ID.

Depending on the manufacturer, preinitialized floppy disks have a volume ID of either all blanks or IBMIRD. You can change these names with this option.

PG

Use this option to purge all the files on the floppy disk and then return all the label sectors to their original preinitialized state. This option accesses only the floppy disk index track sectors 7 through 25; that is, it changes only the floppy disk directory, while all the other sectors remain the same.

The header record length field in all file headers are set to 80 bytes unless you set switch 1 (SW1) at the time that you execute the MDCOPV program. If you set SW1, the program sets the length to 128 bytes for ICMD format floppy disks or 256 bytes for MMDU format floppy disks.

RM

Use this option to remove files on an individual basis. The MDCOPV program does not permit multiple or group removals. When you remove a file from the floppy disk, the system returns the directory header of the file and all the data sectors. The MDCOPV program then tries to combine all contiguous available sectors into one large available area. If no sectors adjacent to the returned area are available, the area is marked available, so possible checkerboarding can occur.

This option can remove a file that contains invalid extents. The MDCOPV program does not try to combine this space with any other available areas. The file header is written with the extents equal to their preinitialized values. Removing a file in this manner can cause a missing disk situation.

The header record length field in the removed file header is set to 80 bytes unless you set switch 1 (SW1) when you execute the program. If you set SW1, the length is set to 128 bytes for ICMD format or 256 bytes for MMDU format.

IN

Use this option to initialize (format) a floppy disk. The process of initializing (formatting) floppy disks destroys all data previously recorded on the floppy disk. You can use this option to make an unreadable floppy disk readable.

You cannot use the MDCOPV program to initialize ICMD-LAK format floppy disks. You must use the DIAG20 program, which is released on the maintenance test program tape.

<unit number 0–3>

Use this option to indicate the unit number of the device. Because of hardware limitations, the unit numbers must be 0 or 1 when you are using an HT-DLP, or 2 or 3 when you are using a UC-DLP.

<channel number>

Use this option to indicate the channel number of the floppy disk drive you are using.

<input file name>

The MDCOPV program requires this option only for the CH, RM, and RN options. Use this option to indicate the name of the file to change (CH) or remove (RM), or to indicate the volume ID to rename (RN).

<output file name>

The MDCOPV program requires this option only for the command that changes the name of a file (CH). Use this option to indicate the new name of the file, which can be any name from 1 to 8 characters in length. The MDCOPV program ignores a multifile name entry when you are using the CH function.

BYPASS

Use this option only for the commands that remove a file, or change its name.

This option enables you to remove or to change a file name that has its bypass indicator set. If the file you are accessing has the bypass indicator set and you do not use this keyword, a "file-not-on-floppy-disk" condition occurs.

Note: *The bypass indicator cannot be set or reset by the CH function.*

I<01–13 interlace>

Use this option only for the IN function. It specifies the data pattern to be written during the floppy-disk initialization. Specify the interlace code by entering *I* followed by the 2 digit interlace number, which can range from 01 through 13.

The default interlace code for a floppy disk connected through a DLP (the MMDU format) is I13. The default for a floppy disk connected through a standard I/O control (the ICMD format) is I01.

"

You must use quotation marks to end a command string that you enter for the MDCOPV program.

Copy Function for Floppy Disk Image Files

An image file is a single file on a 100-byte disk that contains all the information and files for a floppy disk. In some cases, image files are included on release tapes and can be copied to floppy disk.

To perform a copy of the whole floppy disk image file to or from the floppy disk, enter the syntax for the MDCOPV program, as shown in Figure 14–4.

In the syntax, use the <number-of-copies 00–99> option to specify where the image file is going. A value of 00 writes the file to a 100-byte disk. Any other value causes the MDCOPV program to read an image file on disk and copy it to floppy disk, and to repeat this process the specified number of times.

Unisys Customer Service Engineering representatives use this feature.

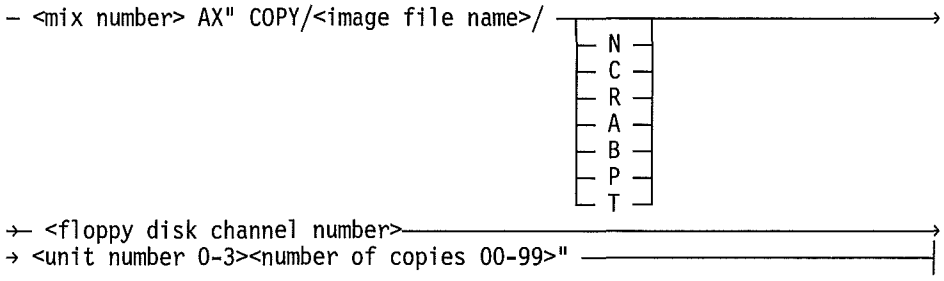


Figure 14–4. MDCOPV Floppy Disk Image File Syntax

<mix number> AX "

You should have written down the mix number when you executed the MDCOPV program. Enter the commands for the program with the AX command. The commands for the MDCOPV program must be enclosed in quotation marks.

COPY/<image file name>

Use this option to indicate the name of the file to copy. It can be from 1 to 6 characters long.

N

Use this option to indicate that the MDCOPV program should not check the part number.

C

Use this option to initiate a compare pass after writing each copy.

R

Use this option to recycle used floppy disks for new copies.

A

Use this option to set the options N, C, and R.

B

Use this option to set the options C and R.

P

Use this option to protect the floppy disk master file or copies (inserts P in headers, includes option C).

T

Use this option to terminate the program without removing the floppy disk (one copy, includes options N, C, and R).

<floppy disk channel>

Use this option to indicate the channel number of the floppy disk drive.

<unit 0–3>

Use this option to indicate the unit number of the floppy disk drive.

<number of copies 00–99>

Use this option to indicate the number of copies you want. Enter a two digit number to indicate the number of copies.

If you enter 00, the MDCOPV program copies the file from the floppy disk to 100-byte disk.

"

You must use quotation marks to end a command string that you enter for the MDCOPV program.

Conversion Functions

The MDCOPV conversion function converts 100-character-per-record code files or 128-character-per-record S-code files into a 256-character-per-record disk or disk pack file. You can use the conversion function to copy code files to a 5.25 inch floppy disk, or you can use the copy function.

Use the MDCOPV conversion function to convert the following types of files:

- An object code file created by the BPL compiler with the MCPP compiler option set
- An object code file created by the ASMBLR compiler
- A 128-character-per-record S-code file created by the PTL compiler on an A Series System

The syntax shown in Figure 14–5 starts the floppy disk file conversion function.

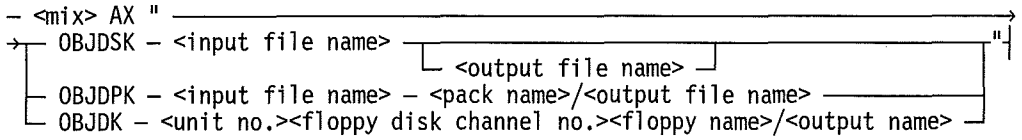


Figure 14–5. MDCOPV File Conversion Syntax

<routine type>

Use this option to specify a 6-character concatenation of the literal OBJ and a 3-character output device mnemonic. The following values are the only ones Unisys supports:

- **OBJDSK.** The object or S-code file is converted to a 256-character-per-record format on disk.
- **OBJDPK.** The object or S-code file is converted to a 256-character-per-record format on disk pack.

- **OBJDK**<unit number>. The object or S-code file is converted to 256-character-per-record format and then copied to the floppy disk on the specified unit number. You must specify the channel number of the floppy disk drive for this option.

<floppy disk channel number>

Use this option to specify the channel number of the floppy disk drive. The MDCOPV program requires this number if you use the routine type OBJDK; otherwise, it is invalid.

<input file name>

Use this option to identify the input object or S-code file. You can specify a multifile ID if the file is on disk pack.

<output file name>

Use this option to specify the name of the output file. You can specify a multifile ID for disk pack and floppy disk files. The MDCOPV program requires this option for output to floppy disk, but does not require it for disk or disk pack output files. The default output file name is the same as the input file name, except that the last character of the name is a z.

<pack name>

Use this option to specify the name of the pack that you want the file to be copied to.

<floppy disk name>

Use this option to specify the name of the floppy disk that you want the file to be copied to.

Examples

The following examples show how to enter commands to the MDCOPV program.

Copying files to a floppy disk:

```
31 AX "DSKDK2 24 HSTLDP HSTLDP"

31 AX "DSKDK3 13 HSTLDQ SYSTEM/HSTLDQ BYPASS"

31 AX "DPKDK3 13 HERBIE/HSTLDN MINI/HSTLDN BYPASS"
```

Copying files from a floppy disk to disk and disk pack:

```
31 AX "DK2DSK 24 HSTLDP HSTLDP "

31 AX "DK3DSK 13 SYSTEM/HSTLDQ HSTLDQ BYPASS"

31 AX "DK3DPK 13 MINI/HSTLDN HERBIE/HSTLDN PSEUDO BYPASS "
```

Copying an entire floppy disk to another floppy disk:

```
31 AX "DK2DK3 7 7"

31 AX "DK2DK3 7 C99"

31 AX "DK3DK2 7 7 C32"
```

Copying image files to and from a floppy disk:

```
31 AX "COPY/PANDLP/A13300"

31 AX "COPY/PANEL1/C13201"
```

Converting and copying code files to floppy disks that can be executed in control state:

```
31 AX "OBJDSK LOADFW LOADFW"

31 AX "OBJDPK LOADFW HRWPAK/LOADFW"

31 AX "OBJDK2 0 LOADFW CV330A/LOADFW"
```

Analyzing files:

```
31 AX "PD3 13"

31 AX "PN3 13"

31 AX "PS2 13"

31 AX "CH2 12 MINI/FINBAR HOMER BYPASS"

31 AX "RN2 12 MYMINI"

31 AX "PG3 12"

31 AX "RM2 13 MINI/HOMER BYPASS"

31 AX "IN3 15"
```

Converting files:

```
31 AX "OBJDK3 10 M68FVR PTDISK/M68FVR"

31 AX "OBJDSK LOADOB LOADFW"
```

File Equate Information

Table 14–1 lists the internal and external file names that the MDCOPV program uses:

Table 14–1. Names of Files for MDCOPV

Internal Name	External Name	Comment
MDSTAT	MDSTAT	Status Report Listing
PAKFIL	PAKFIL	Output Disk/disk pack file
PSRFIL	#<file number>	Pseudo card reader file
INPAK	INPAK	Input Disk/disk pack file
FLCFIL	FLCFIL	Image file - ICMD format
COPFIL	COPFIL	Image file - MMDU format
Object Code input	OBJINP	OBJINP
S-code input	SCODEI	SCODEI
Converted Object Output	OBJOUT	OBJOUT

MDCOPV Messages

The messages the MDCOPV program displays fall into these categories:

- **Informational messages.** These messages indicate a function is complete and requires no operator intervention.
- **Input prompts.** These messages prompt you for some type of input.
- **Error messages.** These are error messages that do not require operator intervention. After displaying one of these messages the MDCOPV program usually aborts the current operation and returns to the "waiting-input-parameters" state.

Informational Messages

The following messages provide information; they do not require a response. However, you may need to act on the information in the message.

`<file name> CHANGED TO <file name>`

The specified file name was changed successfully to a different file name with the CH function.

`<volume ID> CHANGED TO <volume ID>`

The specified floppy disk volume ID was changed successfully to a different volume ID with the RN function.

`<file name> COPIED TO MINIDISK AS <volume ID>/<file name>`

A disk or disk pack file has been copied successfully to the floppy disk identified by <volume ID>/<file name>.

```
<volume ID>/<file name> COPIED TO <media> AS <file name>
```

The floppy disk file identified by the <volume ID>/<file name> construct has been copied successfully to either disk (<media> DSK) or disk pack (<media> DPK).

```
MINIDISK COPY COMPLETED ON <channel number/unit number>
```

The duplicate floppy disk function has been completed successfully.

```
MINIDISK COPY #<copy number> COMPLETED ON <channel number/unit number>
```

The duplicate floppy disk function has completed copy number <copy number>. Remove the floppy disk on the specified channel number and unit number and replace it with a scratch floppy disk if you want more copies. Then enter <mix> AX GO to make more copies.

```
<channel number/unit number> MINIDISK PURGED
```

The floppy disk on the specified channel number and unit number has been purged successfully.

```
<file name> REMOVED FROM MINIDISK
```

The floppy disk file <file name> has been removed successfully from the floppy disk.

```
<channel number/unit number> VOLUME LABEL ID<volume ID>
```

This message comes from the PN function and shows the volume ID of the floppy disk.

```
<interlace type> MINIDISK INITIALIZED ON <channel number/unit number>
<bad tracks message>
```

These messages appear after a floppy disk has been initialized successfully. The interlace type will be either INTERLACED or SEQUENTIAL, depending on the interlace code used during initialization. The bad tracks message will be one or more of the following:

- NO BAD TRACKS
- FIRST BAD TRACK = nn
- SECOND BAD TRACK = nn

Input Prompts

When the MDCOPV program displays the following prompts, you must respond with an AX command before the program can continue processing.

```
ADDITIONAL <memory required> KD REQUIRED FOR READ BUFFER REEXECUTE MDCOPV WITH
ADDITIONAL MEMORY
```

Because of the size of the input/output file buffer, the program requires additional memory for processing this function. Enter *EX MDCOPV MEMORY + <memory required>* to reexecute the MDCOPV program.

```
CONTINUATION MINIDISK <sequence number> REQUIRED LOAD AND <MX>AXDK<UNIT#>
```

The program is reading or writing a floppy disk file that extends across more than one floppy disk and requires the next floppy disk in sequence. If you are creating a file, the sequence number is the number that is written to the floppy disk identified by DK<UNIT#>. If you are reading the file, the sequence number identifies the next floppy disk required. Load the appropriate floppy disk and enter the appropriate ODT response.

```
ENTER INPUT PARAMETERS FOR MDCOPV
```

The MDCOPV program displays this message when it has been executed. When you have entered the input parameter string, and the specified function is complete the MDCOPV program displays this message again and waits for input. Enter *<MIX>AX END* to end processing.

```
INVALID R/D RETURNED ON SECTOR WRITE DIR R/D: <result descriptor> DIR EXT R/D:
<extended descriptor> ENTER R RETRY OR N NO RETRY
```

An attempt was made to write the word *deleted* on a faulty sector. The result descriptors that appear here were received after 10 retries. At this point, enter N. The MDCOPV program stops the current function and returns to the waiting input parameters state.

```
MINIDISK COPY #<copy number> COMPLETED ON <channel number/unit number> MOUNT A
SCRATCH MINIDISK ON <channel number/unit number> AND ENTER GO WHEN READY
```

The duplicate floppy disk function has completed copy number <copy number>. Remove the floppy disk on the specified channel number and unit number and replace it with a scratch floppy disk if you want more copies. Then, enter *<MIX> AX GO* to make more copies.

```
READ EXTENDED FAILED R/D DIR R/D: <result descriptor> DIR EXT R/D: <extended
descriptor> ENTER R = RETRY OR N = NO RETRY
```

The result descriptor and extended descriptor that appear here were received when the MDCOPV program attempted to perform a Read Extended Status operation. An invalid result descriptor was received on the previous I/O, which caused the MDCOPV program to attempt a Read Extended Status operation to determine the exact problem. Enter *N* to have the MDCOPV program abort the current function and return to the "waiting input parameters" state.

```
RECYCLE USED MINIDISK OK? ENTER Y(ES) OR N(O)
```

The floppy disk to be used for making a copy of an image file contains valid data. If it is all right to overwrite this floppy disk, then enter <MIX> AX Y; otherwise enter <MIX> AX N.

```
INPUT REC SIZE EXCEEDS <record size> BYTES TRUNCATION WILL OCCUR IF COPIED TO
MINIDISK CONTINUE Y = YES N = NO
```

Caution

This message indicates that the disk or disk pack file that is being copied to the floppy disk has a record size greater than 128 for ICMD format or 256 for MMDU format. If the copy is allowed to continue, the records written to the floppy disk will be truncated and data will be lost.

```
INVALID RD RETURNED ON <channel number/unit number><result descriptor> ENTER R
RETRY, OR N NO RETRY
```

An I/O error occurred while the MDCOPV program was reading or writing to the floppy disk on the designated channel number and unit number. If you enter N, the MDCOPV program displays the following messages and waits for input:

```
COPY TO MINIDISK ON <channel number/unit number> FAILED
MOUNT A SCRATCH MINIDISK ON <channel number/unit number>
AND ENTER GO WHEN READY
```

Error Messages

After the MDCOPV program displays one or more of the following error messages, it aborts and returns to the waiting input parameters state.

ABORTED DUE TO BAD SECTOR

This message appears if you enter a NO response to the message ENTER R = RETRY OR N = NO RETRY. A memory dump occurs when this message appears.

<file name> ALREADY PRESENT ON MINIDISK

The disk or disk pack file to be copied to the floppy disk already exists. Either copy the file to another floppy disk or remove the file from the floppy disk with the RM function.

BAD FILE-UNDIGITS IN START ADDRESS

The object code file input to the conversion function has undigits in the start address at BASE.+94. You cannot use this file.

BAD FILE-No Start Address at 94

The object code file input to the conversion function has zero in the start address at BASE.+94, or the address controller at the start address is not zero. You cannot use this file.

DUPLICATE MEDIA NOT ALLOWED

The six characters that indicate the direction of the copy cannot contain duplicate hardware types.

FILE ID NOT CHANGED

The file name specified for this CH function already exists on the floppy disk.

INPUT FILE NOT ON MINIDISK

The input file specified cannot be found on the floppy disk. The file name and bypass indicator must match before you can access this file.

INV CONTINUATION FILE SEQ#, IS nn, EXPECTED nn

The MDCOPV program encountered a multivolume floppy disk file, and you put in the wrong continuation floppy disk. The sequence numbers that appear here are the ones that MDCOPV expected and the ones actually received.

INV HEADER EXTENTS, SECTOR number

The File Header number that appears here has overlapping or invalid extent fields. The MDCOPV program cannot use this floppy disk until you perform a purge or initialize it.

INV INPUT BLOCK SIZE > 99

The records-per-block value in the disk file header of the input disk or disk pack file is greater than 99. If you want to copy this file to the floppy disk, it must be reblocked.

INV MULTI VOL SEQ# <header-volume-seq-number>

The header-volume sequence number found in the input File Header label is invalid. MDCOPV expects either a blank volume sequence number and blank multivolume indicator for single floppy disk files, or a C or L in the multivolume indicator and 01 through 99 in the volume sequence number for multivolume floppy disk files.

INV OUTPUT NUMBER OF COPIES

The MDCOPV program expected the number of copies parameter, or the parameter was invalid in some way. The correct syntax is C followed by a 1- to 3-digit number. There is no space between the C and the number (for example, 'C10' is the correct entry if you need 10 copies).

INVALID CONTINUATION MINIDISK REC SIZE

The record size that appears in the File Header of a continuation floppy disk does not equal the original record size.

INVALID IMAGE-COPY ROUTINE INPUT ENTERED

The MDCOPV program cannot find the name of the image file or the second '/' delimiter.

INVALID INPUT ENTERED

The string you entered in response to the prompt ENTER INPUT PARAMETERS contains an invalid token. An example of an invalid token is a multifile name that is greater than 6 characters.

INVALID INPUT FILE RECORD SIZE

The code file that you designated for the conversion function does not have a record size of 100 characters (object code) or 128 characters (S-code).

INVALID INPUT FUNCTION

The input hardware type you specified is invalid, or the MDCOPV program does not recognize the 3-character analyze function you have entered.

INVALID INPUT ROUTINE TYPE

The first item in the input parameter string is not 3 characters (for the analyze function), 6 characters (for the copy function), or 'COPY/' for the image file copy function.

INVALID INTERLACE CODE

The interlace code that you entered as part of the IN function parameter string was invalid. The correct format is Inn where nn represents a number in the range of 01 through 13.

INVALID MINIDISK CC/U

The channel and unit combination you have entered is not correct.

INVALID MINIDISK CHANNEL #

The channel number you have entered either is not present or is invalid.

INVALID MINIDISK UNIT #

The unit number you have entered is not in the range of 0 through 3.

INVALID OUTPUT FILE ID

The MDCOPV program expected an output file name but did not find one.

INVALID OUTPUT FUNCTION

The output hardware type you have specified is invalid.

INVALID OUTPUT REC SIZE

The record size value in the label for this floppy disk file is not mod 4. The MDCOPV program aborts the copy function.

INVALID '(O)CCUNN' INPUT

The parameters for the image file copy function were missing or invalid.

INVALID S-CODE FILE FORMAT

The input S-code file has a record size of 128 characters, but does not contain PTL in the first record. Therefore, this S-code file has an incorrect format.

I/O ERROR IN TRACK ZERO BAD MINIDISK - INITIALIZE ABORTED ON <channel number/unit number>

MDCOPV was unable to initialize the floppy disk because of an error in track zero. The floppy disk is not usable and you should discard it.

MINIDISK MEDIA NOT PRESENT

A floppy disk hardware specification (DK<unit number>) was not found in the routine type you specified. A floppy disk must be present as either the input or output device.

NO AVAILABLE SPACE ON MINIDISK

There is insufficient room on this floppy disk to complete the requested function. You should either use another floppy disk or obtain space on the current floppy disk by removing files or by purging the floppy disk.

NO CHANGED TO FILE ID

A CH function was requested, but you did not enter an output file name.

NO FILES ON MINIDISK FILE <file name> NOT LOADED

An image file could not be created because there were no files on the floppy disk.

NO INPUT FILE ID PROVIDED

You must enter a file name for the CH, RM or RN functions.

NO RECORDS IN INPUT FILE

The file to be copied to the floppy disk contains no records. Your request will be aborted.

OUTPUT MINIDISK NAME IS REQUIRED

A floppy disk name must be input when converting an object code or S-code file directly onto a floppy disk.

VOL ID MISMATCH, MINIDISK ID <volume ID>

The volume ID of the floppy disk does not match the multifile name you entered for the floppy disk.

WRONG FILE (NOT MCPP OR ASSEMBLER)

This code file cannot be converted for one of the following reasons: This object code file was not compiled with the ASMBLR compiler. This object code file was compiled with the BPL compiler, but you did not set the MCPP dollar option.

WRONG MINIDISK FORMAT FILE <file name> NOT LOADED

This floppy disk cannot be used to make an image file.

1 TO 6 CHARACTERS FOR IMAGE FILE NAME

The image file name was missing or was greater than 6 characters in length.

Section 15
MERG:—MCP Merge Intrinsic Program

Overview

The MCP Merge Intrinsic (MERG:) utility program is the only MCP merge intrinsic program that you can use with COBOL ANSI-74. This section contains an overview of the MERG: program and a concise functional description. If you are interested in a complete discussion of this program, you should refer to the *V Series COBOL ANSI-74 Compiler Programming Reference Manual (Volumes 1 and 2)*.

When you invoke a merge, the MCP initiates a MERGE task to perform the merge and suspends the calling program until the merge is complete.

The MERG: program uses information from the input parameters to establish how much memory is required for the merge operation. First, one buffer is allocated for each input file; the buffer size is determined with the following formula:

```
file record length x blocking factor
```

If you specified any alternate areas for a file in the COBOL program, then the MERG: program will allocate a second buffer for this file.

At this point, the MERG: program determines its own memory requirement.

- If it needs more memory than the COBOL program is using, the MERG: program uses a GROW BCT to acquire more memory. This extra memory is returned to the MCP when control is returned to the COBOL program.
- If the MERG: program finds that the COBOL program is larger than the calculated merge memory requirement, the MERG: program uses the extra space to allocate a second buffer to any input file for which a second buffer has not already been allocated.

Thus, two input file buffers are allocated if required. However, two buffers are always allocated for the output data file.

Functional Description

The MERG: program accepts presorted input files and merges them together to create one output file. The MERG: program can merge up to eight input files into a single output file. The physical block size of input and output files can be different. However, the files to be merged must all have an identical record size.

Records are merged on the basis of the ascending or descending keys that you specify in the COBOL MERGE statement. The rules for comparing operands in a relation condition are used to compare corresponding key record fields of the input files that you want to merge. When these key fields are equal, the order of the output of these records follows the order in which you name the files with the USING clause of the COBOL MERGE statement. All such "equal" records with one input file are output before records from another file.

The MERG: program assumes that you have ordered all input records identically. If the MERG: program does not find input records in the sequence you specified with the merge keys, these improperly ordered records are output immediately after being read. The MERG: program continues to output data records from an out-of order file until it finds a record that brings the file back into the proper sequence. This procedure can cause the output to be in a different ascending or descending sequence from the one anticipated by the program that invoked the MERG: program.

Note: *The MERG: program does not display a warning message when merging files contain incorrectly sequenced keys.*

Section 16
NIFMRG—DCP Utility Program

Overview

Capabilities

The NIFMRG utility program enables a data communications subsystem to use more than one type of data communications processor (DCP) simultaneously. The DCPs can be Unisys B 874s, B 974s, CP 2000 communications processors (ICPV), an Offload Reader Sorter Data Link Processor (ORS-DLP), and a Telcom DLP (which are both referred to as DCDLPs).

The ORS feature for the NIFMRG utility enables you to run more than 10 Offload Reader Sorters simultaneously on a V series host. In addition, you can use DCP numbers that are greater than 9 for the ORS type.

Files

The following two Network Information Files (NIFs) are required to define a network configuration for a V Series host:

- Master Control Program Network Information Files (MCPNIF)
- Message Control System Network Information Files (MCSNIF)

The NIFMRG utility program combines the MCPNIF and the MCSNIF files and produces a single MCPNIF file and a single MCSNIF file. After the merge, both of these files will contain information regarding all of the DCPs. The NIFMRG utility alters B,874 firmware files during the merge.

The Network Definition Language (NDL) compilers produce a pair of NIFs that are to be merged for all the DCP devices except the ORS-DLP. Only this pair of corresponding NIFs should be used as input files for the NIFMRG program. The CPNDL1 compiler creates NIFs for B 874s, the NDL 974 compiler creates NIFs for the B 974s, and the NDLDC1 compiler creates NIFs for Telcom DLPs. In addition, the CP2NIF program creates NIFs for each CP 2000 communications processor.

For the ORS-DLP, Unisys supplies four pairs of base NIFs that contain ORS-DLP information to serve the ORS. Each pair of base NIFs, RSPNnF AND RSSNnF, contain information for a particular system number n, where n is either 0, 1, 2, or 3. RSPNnF and RSSNnF will abide by the following naming conventions:

```
STATION NAME:    RDSndd

MCS NAME:        DLPndd
```

The letter "n" represents the processor number for a multi-host environment. The letters "dd" represent the designated ORS DCP number.

One NDL compilation includes information on multiple DCPs of a single type, but it does not contain information on more than one type of DCP.

If all the DCPs are the same type, you can use multiple DCPs in a data communications subsystem without using the NIFMRG utility program. However, if you are using CP 2000s you must use the NIFMRG utility. For further information regarding the NIFMRG utility and NIFs, refer to the *B 974 Network Definition Language (NDL) Programming Reference Manual*, the *B 874/B 974/ORS-DLP MCS Message Headers Programming Reference Manual*, and the *V Series TELCOM DLP Installation and Operations Guide.*

Initiating NIFMRG

You can initiate the NIFMRG utility from the operator display terminal (ODT), or from cards. The following pages explain how to input parameter syntax from the ODT, cards, or a preexisting disk file. In addition, an example of how to compile data in a CANDE data file and submit it to the NIFMRG utility is illustrated.

From the ODT

To enter parameter syntax from the ODT, you must initiate the program by entering the following statement from cards or from the ODT:

```
EX NIFMRG (,"SPO")
```

The NIFMRG program displays the following message:

```
===> ENTER NEXT INPUT <===
```

You can now enter parameter syntax with the AX command. The program flags syntax errors so you can correct the syntax. The input is freeform. Refer to "Statements Longer Than One Line" later in this section for more information.

From Cards

To enter parameter syntax from cards, you must initiate the program by entering the following command from cards or from the ODT:

```
EX NIFMRG
```

The program now looks for a card deck containing the parameter syntax input.

From a Preexisting Disk File

To enter parameter syntax from a preexisting disk file, you must initiate the program by entering the following from cards or from the ODT:

```
EX NIFMRG FILE INPARM=<disk file name> DSK
```

The program now looks for a disk file with the specified name, and the file that contains the parameter syntax. You can create the file with an editor, or in some other manner.

Note: *The disk file must have 80 characters per record and 1 record for each block.*

From a CANDE Data File

You can also create a CANDE data file to compile the data, and submit the data to the NIFMRG utility.

Example

```
?EX     NIFMRG.
?DATA   INPARM.
DCDLP   DDPNOF, DDSNOF, (0,9), (DDONOF, FOONOF),(DDONOF, FO9NOF).
B974    M9PNOF, M9SNOF, (1), (M99NOF, FO1NOG).
ORS     NUPNOF, NUSNOF, (3,17), (RSONOF,FO3NOF),(RSONOF,FO7NOF).
MERGED  PO4NOF, SO4NOF.
END.
?ENDCTL.
```

Parameter Syntax

The following four kinds of statements are used in the NIFMRG syntax:

- DCP
- MERGED
- END
- QUIT

The DCP, MERGED, and END statements are required. You can enter QUIT statements only from the ODT; otherwise, the syntax is the same whether you input it from cards, the ODT, or a disk file. Each statement must end with a period.

DCP Statement

The NIFMRG utility program requires a DCP statement. You should specify one or more DCP statements for each pair of NIFs that you want to merge. The syntax for the DCP statement appears in Figure 16–1.

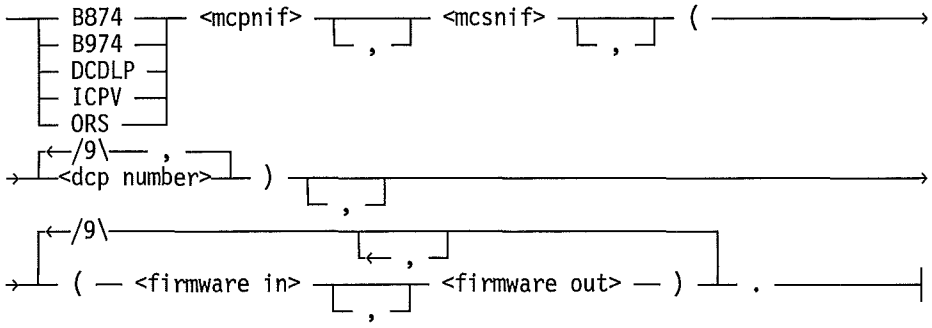


Figure 16–1. DCP Statement

Description

B874
B974
DCDLP
ICPV
ORS

Use these options to specify the type of DCP that will apply to this statement. DCDLP represents Telcom DLPs, or ORS-DLPS if the DCP number is less than 10. ICPV represents CP 2000 communications processors, and ORS represents ORS-DLPs.

<mcpnif>
<mcsnif>

Use these options to enter the file names of the input MCPNIF and MCSNIF files as they are created by the NDL compiler or as they are supplied by Unisys.

<dcp number>

Use this option to enter the numbers of the DCPs that will apply to this statement. A maximum of 10 DCP numbers can be specified in the parenthesis. The merged NIFs contain information only for the DCPs that are declared. DCP numbers are assigned in the NDL source, and must be consistent with the specifications on the DCP UNIT record when you perform a cold-start or a warm-start.

- DCP numbers for B 874s, B 974s, DCDLPs and ICPVs will range from 0 to 9.
- DCP numbers for ORS will range from 0 to 63.

<firmware in>

Use this option to enter the name of the firmware file that is created by the NDL compiler or that is supplied by Unisys. The firmware file is named in the NDL CODE FILE statement. You can change the file name with an NDL file-equate clause.

The names for the <firmware in> files are the exact names of the files as they currently exist. For example, assume that an NDL compilation contains the statement CODE FILE = MC9N0F in the DCP section. In this case, you could use a file-equate clause to rename the code file CC9N0F. The file-equate clause would then cause the compiler to create a code file named CC9N0F. You would need to change this name to MC9N0F before installing the NDL. Otherwise, the MCP would use the CODE FILE name actually stored in the MCPNIF file to identify the firmware file.

When you merge the file using the NIFMRG utility, the <firmware in> file name is CC9N0F. The <firmware out> name can be different; for example, it could be CX9N0F. If you used these names, the NIFMRG utility would create a file named CX9N0F, and the MCPNIF file created by NIFMRG would specify CX9N0F as the firmware name for the DCP.

<firmware out>

Use this option to enter the name of the firmware file that is to be created by the NIFMRG utility. In addition, enter the name that the program will place in the merged MCPNIF file for the firmware file name of this DCP.

Because the NIFMRG utility program creates an output B 874 firmware file that is a modified version of the input, you might want to rename the output file so you have a copy of the original, in case the merge is repeated or abandoned.

You must list the <firmware in> and <firmware out> file pairs in the same order that you list the DCP numbers. For example, if you specify the DCP numbers in the order of 4 followed by 3, the first <firmware in> / <firmware out> file pair must correspond to the DCP number 4 and the second DCP to the number 3.

Constraints

- You can not use merged NIFs produced by the NIFMRG program as input files to create other NIFs.
- The B 874 statement must be the first statement.
- You can not use more than 10 different pairs of NIFs in one input set of DCP statements.
- The NIFMRG utility program will not create an output ICPV firmware file. However, you can specify the same name for the <firmware in> file (default name ICPRUN) and the <firmware out> file, or you can use the system copy command to copy the <firmware in> file as the <firmware out> file manually after the merge
- For B 874s, and B 974s you must specify in the DCP statement all of the DCPs that are included in the input NIF files that are created by the NDL compiler.

MERGED Statement

The NIFMRG program requires the MERGED statement to name the newly merged NIFs. You can specify any number of MERGED statements. However, if you use more than one statement, the NIFMRG program flags the error and processes only the last statement. This enables you to change your mind about the file names. The syntax for the MERGED statement appears in Figure 16–2.

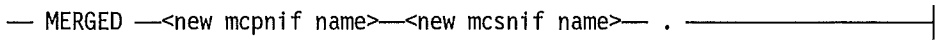


Figure 16–2. MERGED Statement

If the newly merged NIF has the same name as a file that already exists, a DUP LIB (duplicate library file) condition occurs. You can overwrite the old file by entering a mix number with the RM command. If you set the MCP option RMOV, the program overwrites the old files automatically. You can set this option in the system configuration file, or with the SO command and the RMOV option.

Example

```
MERGED CCPNOF CCSNOF.
```

END Statement

The END statement is a required statement, because it signals the end of your input to the NIFMRG program. It must be the last statement you enter in your syntax. If your syntax has no errors, the NIFMRG program begins merging NIFs. If you are entering syntax from the ODT, you can correct the errors and enter END again. The syntax for the END statement appears in Figure 16–3.

Figure 16–3. END Statement

QUIT Statement

Enter the QUIT statement only from the ODT, because it will end the syntax and exit from the program, leaving all of the files unchanged. The syntax for the QUIT statement appears in Figure 16–4.

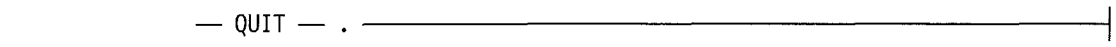


Figure 16–4. QUIT Statement

Syntax Example

Use the following example if you need to create NIFs for a network with B 874 and B 974 communications processors with two ORSs. To add two ORSs to a network, you will need to use a base pair of ORS-DLP NIFs with input parameters that contain the following NIFMRG statements:

```
B874 C8PNOF, C8SNOF, (1), (C8WNOF, FC1NOF).

ORS ORPNOF, ORSNOF, (03,19), (ORFNOF, F03NOF), (ORFNOF, F19NOF).

B974 J9PNOF, J9SNOF, (9), (J09NOF, FC9NOF).

MERGED FCPNOF, FCSNOF.

END.
```

Statements Longer Than One Line

There are times when a DCP statement is longer than the 80-column card image or the 60-column ODT line. All statements can occupy as many lines as needed; just remember to terminate each statement with a period.

Examples

The following examples are DCP input statements that contain more than one line of input:

```
B874 C8PN1F C8SN1F (1,7)
(C81N1F,CC1N2F),(C87N1F,CC7N2F).

B974 C9PN5F C9SN5F
(5)
(C95N1F,C95N2F)
      .
      .
      .
DCDLP CCPN8F,CCSN8F (3,4,2),
(XX3N1F,XX3N2F),
(XX2N1F,XX2N2F).
```

Errors

When NIFMRG finds an error in the syntax, it ignores all input until it finds a period. If the error **is** an omitted period, NIFMRG ignores all input until it finds another period. If the input is from an ODT, NIFMRG flags the error so you can enter the correct syntax.

Listing

The NIFMRG utility program generates a listing that has several parts. A sample listing is shown Figure 16–5.

Description

The first part of the listing labeled "SUMMARY OF NIF-MERGE INPUT PARAMETERS" includes the parameter syntax that you entered, error locations, and the syntax NIFMRG interpreted as single statements. Statements are double-spaced to show where one statement ends and the next begins. The lines for a single statement and its errors are single-spaced.

The next part of the listing is a table labeled "SUMMARY OF NIF-MERGE TABLES GENERATED" that provides information about the MCPNIF and MCSNIF files. This table will include the following information:

- Index numbers for reference

- The names of the original NIF files
- The DCP type
- The total number of DCPs and their corresponding DCP numbers

Following the preceding table is a list of the DCP numbers and the corresponding file names for the firmware in and firmware out files. In addition, the global information count lists the number of valid entries that will be used from the first table and the names of the merged NIF files.

The last part of the listing displays the corresponding station names, logical station numbers (LSNs), MCS IDs, and DCP numbers in the merged NIF files.

```
EXECUTED  ON:  08/18/92  AT  14:56:04     N I F   M E R G E R     R E P O R T     PAGE  001


        SUMMARY  OF  NIF-MERGE  INPUT  PARAMETERS
        ---------------------------------------

B874   NU8NPF,  NU8NSF,  (00)  (NU8NFF,  F00NOF).

B974   NU91PF,  NU91SF,  (01)  (NU91FF,  F01NOF).

ICPV   NU2NPF,  NU2NSF,  (02)  (NU2NFF,  F02NOF).

DCDLP  NUTNPF,  NUTNSF,  (03)  (NUTNFF,  F03NOF).


ORS  NUPNOF,  NUSNOF,  (24,14),  (RSONOF, F24NOF),(RSONOF, F14NOF).

ORS  NUPNOF,  NUSNOF,  (25,15),  (RSONOF, F25NOF),(RSONOF, F15NOF).

ORS  NUPNOF,  NUSNOF,  (05,06),  (RSONOF, F05NOF),(RSONOF, F06NOF).

ORS  NUPNOF,  NUSNOF,  (07,33),  (RSONOF, F07NOF),(RSONOF, F33NOF).


MERGED  PC1NOF,SC1NOF.

END.
```

Figure 16–5. NIFMRG Listing

```
EXECUTED  ON:  08/18/92  AT  14:56:04     N I F   M E R G E R     R E P O R T     PAGE  002

                SUMMARY  OF  NIF-MERGE  TABLES  GENERATED
                ----------------------------------------

INDEX     MCPNIF     MCSNIF     DCP_TYPE     TOTAL_DCPS    DCP_NBRS
-----------------------------------------------------------------
-0--------NU8NPF-----NU8NSF------B874----------01----00.
-1--------NU91PF-----NU91SF------B974----------01----01.
-2--------NU2NPF-----NU2NSF------ICPV----------01----02.
-3--------NUTNPF-----NUTNSF------DCDLP---------01----03.
-4--------NUPNOF-----NUSNSF------ORS-----------08----24,14,25,15,05,06,07,33.
-5--------      -----      ------    ----------  -----
-6--------      -----      ------    ----------  -----
-7--------      -----      ------    ----------  -----
-8--------      -----      ------    ----------  -----
-8--------      -----      ------    ----------  -----

DCP #     FIRMWARE_IN       FIRMWARE_OUT
---------------------------------------
-00----------NU8NFF------------F00NOF---
-01----------NU91FF------------F01NOF---
-02----------NU2NFF------------F02NOF---
-03----------NUTNFF------------F02NOF---
-05----------RSONOF------------F05NOF---
-06----------RSONOF------------F06NOF---
-07----------RSONOF------------F07NOF---
-14----------RSONOF------------F14NOF---
-15----------RSONOF------------F15NOF---
-24----------RSONOF------------F24NOF---
-25----------RSONOF------------F25NOF---
-33----------RSONOF------------F33NOF---

GLBL_INFO_COUNT        MERGED_MCPNIF_NAME         MERGED_MCSNIF_NAME
-----------------------------------------------------------------

-05--------------------------PC1NOF-----------------SC1NOF---

TOTAL OVERALL DCPS DECLARED IS 12.
```

Figure 16–5. NIFMRG Listing (Cont.)

```
EXECUTED  ON:  08/18/92  AT  14:56:04     N I F   M E R G E R     R E P O R T     PAGE  003

          STATION NAME       LSN       MCS ID    DCP #
          ------------       ---       ------    -----

              ST0101         0000      NUOMCS      00
              ST0102         0001      NUOMCS      00
              ST0103         0002      NUOMCS      00
              ST0104         0003      NUOMCS      00
              ST0105         0004      AC          00
              ST0106         0005      NUOMCS      00
              ST0107         0006      AC          00
              ST0108         0007      NUOMCS      00
              ST0109         0008      AC          00
              ST0110         0009      NUOMCS      00
              ST9001         0010      NUOMCS      01
              ST9002         0011      AC          01
              ST9003         0012      NUOMCS      01
              ST9004         0013      AC          01
              ICP000         0014      .PFMCS      02
              ICP001         0015      AC          02
              ICP002         0016      NUOMCS      02
              STT01          0017      NUOMCS      03
              STT02          0018      AC          03
              STT03          0019      AC          03
              STTTTY         0020      TSHMCS      03
              RDS005         0021      DLP005      05
              RDS006         0022      DLP006      06
              RDS007         0023      DLP007      07
              RDS014         0024      DLP014      14
              RDS015         0025      DLP015      15
              RDS024         0026      DLP024      24
              RDS025         0027      DLP025      25
              RDS033         0028      DLP033      33

0000 WARNING MESSAGES

NO ERRORS, FILES MERGED
```

Figure 16–5. NIFMRG Listing (Cont.)

Section 17
PBDPRN—Printer Backup Utility Program

Overview

The Printer Backup Utility Program (PBDPRN) is a bound utility that prints printer backup files on various types of printers, including image page printers. This utility replaces PBDOUT and PBTOUT.

You can use the PBDPRN program to:

- Print printer backup files by number or name from
  - Tape
  - Disk pack
  - Disk
- Create printer backup tapes.
- Search the backup file for a certain string of text and, when found, to print the string.
- Start and stop printing at any position in a file.
- Translate all lowercase letters to uppercase letters during printing.

Notes:

- *PBDPRN can print a printer backup file from disk or disk pack, regardless of the medium on which the file was created. It is possible to have the same backup file number on both disk and disk pack: when this occurs, PBDPRN will print the printer backup file residing on disk. If you want to print the printer backup file residing on disk pack, use the FROM <disk pack name> option.*
- *Prior to MCP/VS 3.2 all files were read block 10, but now PBDPRN can read printer backup files with the same blocking factor as that used to create the file.*
- *The backup number is now included in the file printed message.*

See Figure 17–1 for the PBDPRN syntax.

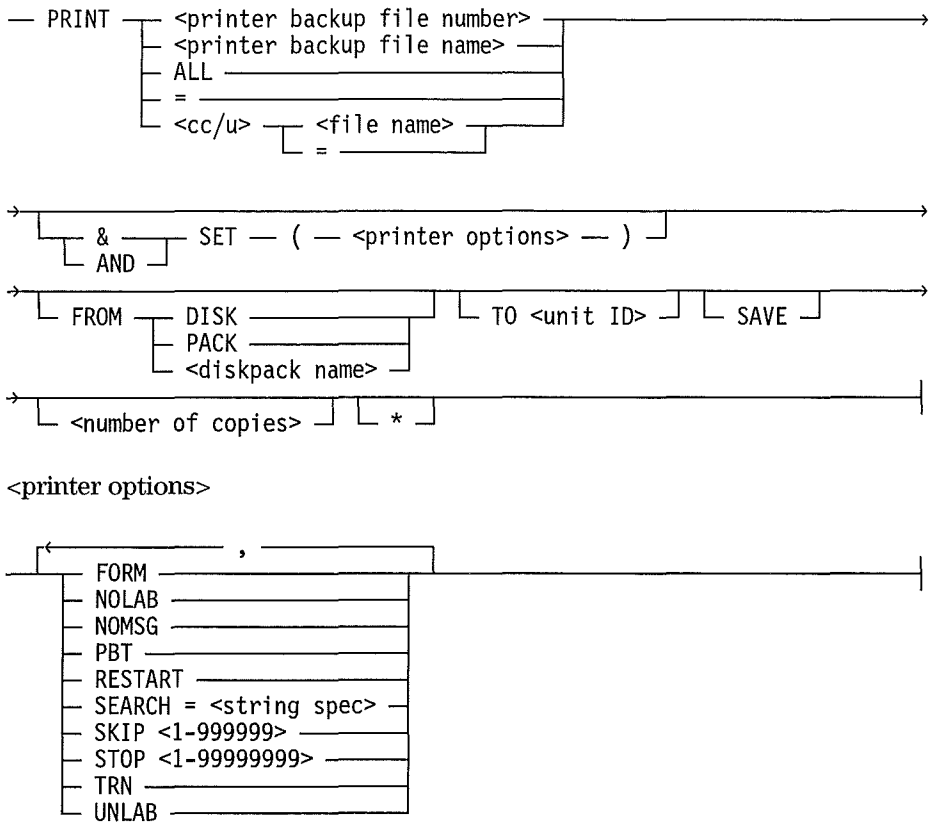


Figure 17–1. PRINT (PBDPRN) Syntax

PRINT

Use this option to invoke the PBDPRN program.

<printer backup file number>

Use this option to print individual files according to their backup file number.

<printer backup file name>

Use this option to print individual files according to their backup file name.

ALL
=

Use these options to print all printer backup files. Unless you specify DISK or PACK, the program prints all the printer backup files on both disk and disk pack.

cc/uu <file name>
cc/uu =

Use this option to print printer backup files from tape. You must specify the channel number and unit number for the tape drive. If you use a file name, only that file is printed. If you enter an equal sign, all the printer backup files that are on that tape are printed.

AND SET
& SET

Use these keywords to set options. Enclose the options in parentheses. If you use more than one option, separate each option with a comma.

FORM

Use this option to direct output to the printer you previously named in the MCP file equate control card (for example, FILE PRINT = SHORT/OUTPUT).

NOLAB

Use this option if you do not want a block letter label to appear on the cover page of a printout. However, the program prints the line that indicates the backup file number, its name, and the file label information.

NOMSG

Use this option to stop the PBDPRN program from displaying messages on the system console when it finishes printing a file.

PBT

Use this option to direct the printer backup listing to tape.

RESTART

Use this option to restart a print job that was stopped with the QT command or terminated along with the PBDPRN program because of VSID message protocol errors. With this option, you can start printing at the beginning of the page the program was printing when it was stopped. If the program does not find the beginning of a page, it resumes at the beginning of the last line it was printing when it was stopped. This option does not work for files on printer backup tapes (PBT).

To use RESTART, enter the same syntax as that you used to start the job originally, but include the option RESTART.

SEARCH <string spec>

Use this option to search a column of text and to start printing the file where it finds a matching string of characters. The string cannot exceed 99 characters.

Figure 17–2 shows the syntax for the SEARCH option:

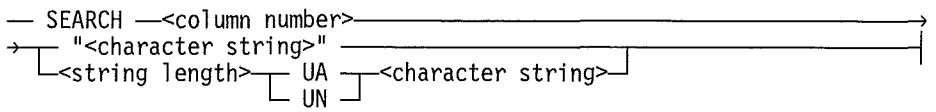


Figure 17–2. SEARCH Option Syntax

<column number>

Use this option to indicate the column to search. It can be any number between 1 and 132. Note that the sum of the column number and the string length cannot exceed 133.

"<character string>"

Use this option to indicate a string of characters that you want the PBDPRN program to search. The string cannot be longer than 99 characters, including blanks.

<string length>

Use this option to indicate how many characters and blanks are in the string. There can be up to 99 characters in the string.

UA
UN

Use UA to indicate that the string is a representation of bytes as you would see on a printed listing.

Use UN to strip the zone portions off a character. You can do this only for a backup on tape condition, where you are printing a backup from DISK/PACK. In this case, the program assumes that UA is the compare type.

SKIP <1–999999>

Use this option to begin printing at a certain line number. Use it with the option STOP to print part of a file.

STOP <1–99999999>

Use this option to stop printing at a certain line number. Use it with the option SKIP to print part of a file.

TRN

Use this option to translate all lowercase letters to uppercase letters.

Note: *This option is automatically set whenever the printer is declared UPPERCASEONLY.*

UNLAB

Use this option when you do not want a label or any other type of information to appear on the cover page of the printout.

FROM DISK
FROM PACK
FROM <disk pack name>

Use the option FROM DISK to specify a printer backup file on disk.

Use the option FROM PACK to specify a printer backup file on disk pack. The PBDPRN program searches the following:

- The primary backup family declared on the CONTROL BACKUP record during system initialization
- The secondary backup family declared on the CONTROL BACKUP record during system initialization (if no secondary backup family is declared, PBDPRN skips this search)
- All unrestricted disk packs

Use the option FROM <disk pack name> to indicate that the printer backup file is stored on a specific disk pack. The PBDPRN program searches only that disk pack for the file.

If you do not include a FROM clause in the PRINT command, the PBDPRN program searches the disk, the primary backup family, the secondary backup family (if any). If no primary or secondary backup families are specified, PBDPRN searches all unrestricted disk packs.

TO <unit ID>

Use this option if you want to specify a particular printer. If you are using an image page printer, you must use the option PCF = <file name>. If you use this option, the file is saved after printing.

If you do not use this option, the PBDPRN program prints the file on the first available line printer.

SAVE

Use this option to save the printer backup file after it is printed.

<number of copies>

Use this option if you want to print more than one copy. You can print up to 99 copies for each print job. If the PBDPRN program is printing backup files from a tape, it ignores this option.

*

Use the asterisk to put the PBDPRN program into a waiting-accept condition. The program will display ENTER SPECS on ODT and PROMPT you with ACCEPT. You can then use the AX command to enter parameters to the PBDPRN program. The use and syntax of the parameters are the same as those used in the PB command. Refer to Volume 2 for more information about the PB command.

Examples

```
PRINT 12345
```

In this example, the printer backup file @12345 is printed on any available line printer.

```
PRINT 12345 AND SET (SKIP 100, STOP 1000)
```

In this example, the printer backup file @12345 prints starting at line 100 and stops printing at line 1000. Because you did not specify a printer, the program printed the file on any available line printer.

```
PRINT 6/1 =
```

In this example, all the backup files from the tape on the drive with the cc/uu number 6/1 are printed on any available line printer.

```
PRINT BKFILE & SET (SEARCH 10 "SAM") TO SPEC 5
```

In this example, the program searched column 10 of the text and began printing when it found the string SAM. The program printed that entire line through to the end of the file. You specified the printer named SPEC, and you requested five copies.

Executing PBDPRN

You can execute the PBDPRN program with the EXECUTE command and the following syntax:

```
EX PBDPRN.
```

When you have executed the program, the following instructions appear on the ODT:

```
PBDPRN = <mix number> ENTER SYNTAX. AX 'END' TO TERMINATE INPUT.
```

Enter syntax with AX commands, and then enter END to terminate your input.

Auto Printing

You can use MCP control file equation cards to automatically direct printer backup files to the image page printer.

The following examples illustrate the different syntax that you use to print automatically to a line printer rather than an image page printer.

To print to a line printer, the MCP FILE control card should look like this:

```
?FILE LINE = SPEC/OUTPUT FORM AUTO
```

To print this file to an image page printer, the card should look like this:

```
?FILE LINE = DOCPCF/OUTPUT FORM AUTO.
```

Note: *DOCPCF is the unit ID for an image page printer.*

PBDPRN Syntax Errors

The PBDPRN program displays the following messages when errors in syntax are input from a terminal.

`AT <token>`

This message indicates where in the syntax an error occurred.

`BAD COMPARE STRING`

Refer to the PB command in Volume 2.

`BAD OFFSET`

Refer to the PB command in Volume 2.

`BAD SKIP FACTOR`

An invalid skip was detected.

`BAD STRING LENGTH`

An invalid length of compare string was detected.

`DESTINATION MISSING`

The keyword TO must be followed by the unit ID of the printer the PBDPRN program is to print the file on.

`INVALID COPY NUMBER, TWO DIGIT NUMBER EXPECTED`

The PBDPRN program prints a maximum of 99 copies for each execution.

`INVALID FILE IDENTIFIER`

File names can be up to 6 characters long.

`INVALID OPTION, 'SET' EXPECTED`

You must use the keyword SET when you invoke the AND SET options. For example, enter the following:

```
PRINT 12345 & SET (NOLAB, TRN).
```

`INVALID SET OPTION OR ')' EXPECTED`

Valid options are listed in the railroad diagram. They must be in parentheses. For example, enter the following:

```
PRINT MYFILE AND SET (NOLAB).
```

```
INVALID SOURCE, 'PACK' OR 'DISK' EXPECTED
```

PACK OR DISK are the proper names for media.

```
INVALID STOP NUMBER
```

The PBDPRN program stops printing files at a line number from 1 to 99999999.

```
INVALID SYNTAX, 'FROM' OR 'TO' EXPECTED
```

You specified something other than FROM, TO, SAVE, copy number, or * after the file name and the AND SET options.

```
MISSING FILE IDENTIFIER
```

The PBDPRN program needs to know the file to print. You can identify files by specifying the backup file name, backup file number, the keyword ALL, or the symbol =.

```
MISSING LEFT PAREN
```

Parentheses are required with the AND SET options. For example, enter the following:

```
PRINT MYFILE AND SET (NOLAB).
```

```
'UA' OR 'UN' EXPECTED
```

When you use the SEARCH option and specify the length of the string, you must indicate if the string is UA or UN. For example, enter the following:

```
PRINT 12345 & SET (SEARCH 1 3 UA SAM).
```

AX Errors

When you use an asterisk at the end of the command syntax, you can then enter various options with the AX command. If you make an error, the PBDPRN program displays one of the following messages.

```
BAD COMPARE STRING
```

Refer to the PB command in Volume 2.

```
BAD OFFSET
```

Refer to the PB command in Volume 2.

```
BAD SKIP FACTOR
```

Refer to the PB command in Volume 2.

```
ENTER SPECS
```

When you invoke the PBDPRN program with an asterisk at the end of the command string, it displays this message to let you know that it is ready for you to enter options with the AX command.

```
INVALID PARAMETER - TRY SOMETHING ELSE
```

The syntax that you entered is not correct. Try again.

Messages Displayed for Line Printers

The program displays the following messages, depending on the printer you are using.

```
<F-N> PRINTED cc/uu COPY [<copy-number>]
```

Refer to the PB command in Volume 2.

```
<F-N> STOPPED cc/uu <copy-number> RECORD <record number>
```

Refer to the PB command in Volume 2.

```
*** PARITY ERROR AX OR DS?
```

The PBDPRN program has detected a parity error in a tape file. Use the AX command to force the program to print error flags marking the corrupt block and to continue printing the file. Use the DS command to discontinue the job.

```
FILE NOT PRESENT
```

The PBDPRN program cannot find the file on disk or disk pack.

```
FILE NOT STOPPED
```

Refer to the PB command in Volume 2.

```
INVALID FILE
```

Refer to the PB command in Volume 2.

```
INVALID PAPERMOTION FIELD
DS OR DP?
```

Refer to the PB command in Volume 2.

```
NO MATCHING RECORD
```

Refer to the PB command in Volume 2.

```
PBDPRN =<mix number> CURRENT PRINT IS <percent> COMPLETE
ESTIMATED TIME TO COMPLETION IS hh:mm:ss (EOJ AT hh:mm)
```

The PBDPRN program displays this message to indicate when a document will finish printing. This message is in response to the SW command. To find out when the PBDPRN program will finish printing, enter the following:

```
<mix number> SW1 1.
```

```
PBK ON MORE THAN ONE SYSTEM
```

More than one file with the same name exists on the shared system when PBDPRN program asks for the system number. Enter the system number with the AX command so the program can continue with the print job.

```
PRINT SKIPPED [FORWARD/BACKWARD] <a number of> RECORDS
```

A number of records have been skipped, forward or backward, because you used the SK command.

```
STRING FOUND AT RECORD <record number>
```

Refer to the PB command in Volume 2.

Section 18
PCOPY—Object Program Copy Utility Program

Overview

The PCOPY program copies and modifies program code files or WFL code files. The PCOPY program transfers code files from a 100-byte disk to a 180-byte disk pack and vice versa.

The disk pack or disk pack family on which the programs reside can be either shared or nonshared. Refer to the CONTROL CODEPACK record in Volume 1 for more information about code files on disk pack.

The PCOPY program does not support the transfer or conversion of remote files through the BNA network.

Operating Instructions

Execute the PCOPY program with the following syntax:

```
EX PCOPY (100 , "<program ID>" , "<disk pack ID>")
```

The 100 is an optional parameter setting VALUE 0 equal to 1.

The program ID that you specify must reside on a 100-byte disk or a 180-byte disk pack and must be a valid code file. The program produces self-explanatory error messages if these criteria are not met.

- If you want to transfer the code file from 100-byte disk to 180-byte disk pack, the disk pack ID must identify the disk pack or disk pack family where the code file is to reside. It must be a valid full disk pack name or family name (not a disk pack group ID). A transfer from 100-byte disk to 180-byte disk pack is the default value for the PCOPY program.
- If you want to transfer the code file from 180-byte disk pack to 100-byte disk, you must use the VA 0 1 option. With this option, you indicate that the destination of the code file is a 100-byte disk, and that the disk pack ID identifies the name of the disk pack or disk pack family where the code file currently resides.

When the code file transfer to disk pack is complete, you can execute it in the normal manner. Refer to the EXECUTE and COMPILE statement syntax in Volume 2 for more information.

Note: *PCOPY is not necessarily needed when you are using the COBOL ANSI-74 and RPG compilers (release ASR 6.5, and greater). These compilers are capable of generating code files directly to disk pack in the proper format. Refer to COMPILE statement syntax in Volume 2 for more information.*

Examples

```
EX PCOPY (,"COBOL", "SPLIBS")
```

In this example, the program transfers the COBOL compiler to the disk pack family named SPLIBS. You can then use the compiler to compile a program with the following statement:

```
CMP JAKE WITH COBOL ON SPLIBS LIB.
```

In this example, the program creates a new program called JAKE on DISK.

```
EX PCOPY (100, "COBOL", "SPLIBS")
```

In this example, the program transfers the code file named COBOL from the disk pack family named SPLIBS to 100-byte disk.

Section 19
PKCOPY—Disk Pack Copy Utility Program

Overview

Use the PKCOPY program to exactly duplicate all the data on one disk pack on a second disk pack. Both disk packs must be the same type and model. The PKCOPY program performs the copy function on a sector-by-sector, direct I/O basis that makes it faster than SYSTEM/COPY. Format the destination disk pack before you use it.

Neither disk pack can be accessed until the copy function is complete. When it is complete, both disk packs are in a saved mode. When a disk pack is in the saved mode, you must use the RY command to make it available.

When the copy function is complete, one disk pack is an exact duplicate of the other. Each disk pack has the same files, the same family name, the same available table, and, depending on whether or not you use the feature, the same serial numbers.

If PKCOPY detects an error during copying, it changes from a block copy mode to a sector copy mode with a higher retry count, to preserve as much data in the block as possible.

If you choose the COMPARE feature, the data copied to the destination disk pack is compared with the data on the source disk pack. If there is a descrepancy, you are given the option to stop or to continue copying.

The utility also has a feature that lets you change the serial number of the destination disk pack. Otherwise, both disk packs have the same serial numbers. On version 1 disk packs, do not change the serial numbers if the source disk pack contains parts of a file that span two or more disk packs, because the disk pack serial numbers are needed to find all parts of the file.

The MCP uses the available table to keep track of the amount of space available on the disk pack and to omit bad sectors from use. The XP command is used to indicate the bad disk pack sectors that cannot be used. The MCP stores the information that you enter with the XP command in the available table. Therefore, the duplicate disk pack, which has a duplicate available table, has the same sectors omitted from use as the source disk pack does.

The default amount of memory allocated to the program when it executes is 42KD. The default buffer size is 36KD. The program uses one hundred sectors for each I/O for copying and fifty sectors for a COMPARE. For larger I/Os and increased speed, use the MEMORY command to provide additional memory.

Operating Instructions

Use the EXECUTE command to execute the PKCOPY program. You enter PKCOPY commands with the AX command as follows:

```
EX PKCOPY; AX "<command>"
```

Figure 19–1 shows the command syntax for the PKCOPY program.

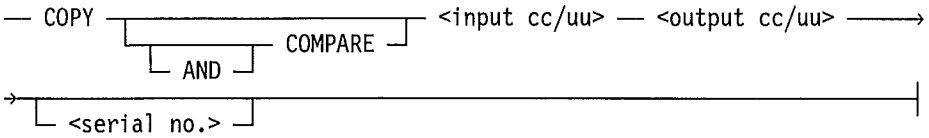


Figure 19–1. PKCOPY Syntax

COPY

This option begins the PKCOPY command syntax. Note that it must begin with a quote.

COMPARE
AND COMPARE

Either option invokes a feature that compares the data on both disk packs. If there is a discrepancy, the PKCOPY program displays an error message and a prompt asking if you want to continue.

<input cc/uu>

This option indicates the channel and unit number of the disk pack from which the data are to be copied.

<output cc/uu>

This option indicates the channel and unit number of the disk pack to which the data are to be copied.

<serial no.>

This option enables you to specify a serial number for the new disk pack. If you do not specify a serial number, both disk packs end up with the same serial number. A disk pack serial number can range from 1 to 999999. On a version 1 diskpack, do not change the serial numbers if the disk pack contains parts of a file that spans two or more disk packs, because the serial numbers are needed to find all parts of the file.

Examples

```
EX PKCOPY; AX "COPY 04/02 04/03"

EX PKCOPY; AX "COPY COMPARE 14/02 04/03 206043"

EX PKCOPY; AX "COPY AND COMPARE 04/02 04/03 206043"
```

Obtaining PKCOPY Status

You can get a report on the progress of PKCOPY while it is executing by entering the following command:

```
<mix number of PKCOPY> SW1=1
```

Example responses follow:

```
CURRENT COPY PHASE IS 25% COMPLETE
ESTIMATED TIME TO COMPLETION IS 00:22:10, (EOJ AT 12:55)

CURRENT COMPARE PHASE IS 80% COMPLETE
ESTIMATED TIME TO COMPLETION IS 00:05:44, (EOJ AT 15:12)
```

The estimated time to completion and the estimated time of completion refer only to the completion of the current phase of the program—COPY or COMPARE. Estimates vary from request to request because PKCOPY is affected by other tasks in the mix and by various I/O factors.

PKCOPY Messages

The PKCOPY program displays messages when it encounters errors while reading or writing. It displays the following message asking whether or not you want to continue:

```
DO YOU WANT TO CONTINUE? (YES/NO)
```

When this message appears on the ODT, use the <mix number> AX command to enter YES or NO. For example, if you want to continue, you might enter the following command:

```
26 AX YES
```

Error messages and explanations for the PKCOPY program appear in the following paragraphs:

```
COMPARE ERROR ON SECTOR address
```

The PKCOPY program copies data on a sector by sector basis. In this case, a compare operation discovered a discrepancy between two sectors that should contain the same data.

INPUT/OUTPUT DRIVE INVALID OR BUSY

The copy operation cannot continue because either one of the two disk packs is in use, or the channel and unit numbers you specified for one or both disk packs are not correct.

INVALID FUNCTION-USE 'COPY' OR 'COPY [AND] COMPARE'

One or more of the commands you entered after the AX command is misspelled or is invalid.

INVALID PACK LABEL <name>

The program did not find the disk pack label <name>.

NO COPY-PACKS MUST BE OF IDENTICAL TYPE

The PKCOPY program copies data between the same types of disk packs. It does not copy data from one type of disk pack to another type of disk pack. You must use the SYSTEM/COPY utility program to do so.

NO COPY-UNKNOWN UNIT ID RECEIVED

The unit number that you specified for one of the disk packs is not correct.

READ ERROR ON SECTOR number UNREADABLE RD = <result descriptor>

A read operation from the indicated disk pack sector did not finish successfully. This result can indicate a bad sector, so that the data it contains could be corrupt.

WRITE ERROR ON PACK <cc/uu> RD = <result descriptor>

A write operation to the indicated disk pack sector did not complete successfully. This can indicate a bad sector, which means that the data it contains could be corrupt.

Section 20
SNPANL—SNAP Analysis Utility Program

Overview

The SNAP Analysis Utility Program (SNPANL) takes a raw SNAP file on a V 500 system and breaks out the machine state for analysis by support personnel.

Obtaining a Hard Copy of the SNAP Picture

To obtain a hard copy of the SNAP picture on a V 500 system, perform one of the following procedures:

- Enter an LNM command to execute MLGOUT and invoke SNPANL. The LNM command transfers and analyzes the maintenance log. SNPANL analyzes any SNAP files that are present.
- Execute SNPANL with a file equate to the specific SNAP picture, as follows:

  ```
  EX SNPANL FILE SNAPFI (. MpSxx)
  ```

  where:

  p = system number

  xx = picture number; value 00 through 99

Either of these steps produces a printer backup file that contains formatted images of the specified SNAP picture.

Section 21
SORT.—Sort Intrinsic Program

Overview

The Sort Intrinsic Program (SORT.) provides a way to execute rapid disk or disk pack sorts from any program written in COBOL ANSI-68 or in BPL.

When you invoke a sort, the MCP initiates a SORT task and suspends the calling program until the sort is complete. For information about the memory used by SORT. refer to "Memory Requirements" in this section.

You can use a parameter file to modify the SORT intrinsic parameters to match the specific requirements of your site. Use a file with the reserved name of SORT.. or SORT.n (where n is the system number) for this purpose. If the operating system finds either of these files when SORT. is executed, it uses the parameters in the file.

Default SORT. parameters direct work files to 100-byte disks, where they are assigned on a space-available basis.

You can also substitute a user-provided SORT. program. The operating system searches the code path before calling the bound SORT. intrinsic. If the system finds any file or program with the file identifier SORT., it ignores the bound version and executes the program.

Memory Requirements

Memory requirements for the SORT. program are the following:

- Minimum limit is 15K digits
- Maximum limit is 300K digits

Within these minimum and maximum limits, the operating system examines the following:

- The value of the LIMIT SRTMEM record in the system configuration file, discussed in the *V Series Systems Operations Guide Volume 1: Installation.*
- The memory occupied by the program that called the SORT. program

The operating system gives the SORT. program the larger of these two values, as long as the amount of memory does not violate the minimum or maximum limits.

The minimum memory limit is larger if files have record or block sizes that approach the maximum limits for record or block size.

To determine the memory requirement of an object program that uses the SORT. intrinsic, use the information contained in the following file information block (FIB) fields as a guide.

- FIBMRL—Record length is specified in digits.
- FIBMBS—Block size is specified in digits.
- FIBVAR—Most common record size for variable-length record; defaults to FIBMRL (maximum record length) if you do not specify variable-length records.
- MM—The size of this field is 100 if FIBMRL is greater than 1000; otherwise it is 20.
- SIZE—For fixed-length records, this field contains the value of FIBMRL rounded to the next highest multiple of MM if FIBMRL is not already a multiple. For variable-length records, this field contains the value of FIBVAR rounded to the next highest multiple of MM if it is not already a multiple.
- TOTAL KEY—Size of all keys is specified in digits.

Using the values from these fields, the minimum memory for the SORT. program is the following:

- Fixed-length records:

  ((2 x FIBMBS) + (10 x SIZE) + (10 x TOTAL KEY) + 7000 digits) or 30,000 digits, whichever is larger

- Variable-length records:

  ((2 x FIBMBS) + (10 x SIZE) + (10 x TOTAL KEY) + 10,000 digits) or 30,000 digits, whichever is larger

Disk Requirements

The SORT. intrinsic uses 100-byte disk for scratch work files unless you specify input parametric information for disk pack (180-byte). The work files used for sorting are about two and one quarter to two and a half times the size of the input file.

Input

The input file must be wholly contained on a single hardware type. You cannot mix hardware types within the same file. For example, a tape file cannot be contained on two different types of tape (MPE, GCR, and so forth). The SORT. program accepts input from any of the following media:

- Cards (coded in BCL or EBCDIC)
- Magnetic tape
- 100-byte disk
- 180-byte disk pack

The following limits apply to SORT. input:

- Maximum blocking factor is 999 records for each block
- Maximum record length is 4,998 bytes
- Maximum block size (record size times blocking factor) is 4,998 bytes

When input records are larger than output records (from SORT.), the input records are truncated on the right to match the smaller size.

For input files on any media other than disk or disk pack, the number of records to be sorted must pass from the calling program to the SORT. intrinsic. The SORT. intrinsic assumes this value is accurate. For disk or disk pack input files, the SORT. program identifies the number of records from the disk (or disk pack) header information.

Additional information required by the SORT. intrinsic includes the following:

- Disposition of input and output files (type of close)
- Instructions regarding disposition of unreadable blocks in the input data file
- Choice of 100-byte disk or 180-byte disk pack for work files
- Instructions regarding sort keys
- Instructions to execute sort and translate if required (refer to Section 13, "MAKTRN—Translation File Generator Program")

Sorting

The SORT. program arranges records from the input file according to the keys that you specify within each record, in the stated ascending or descending collating sequence. Signed numeric keys are sorted algebraically. Negative numbers are less than any positive number.

The following list shows the restrictions on the sort key:

- Sort keys can be mixed, ascending or descending.
- Maximum number of sort keys is 40 (subject to any restrictions imposed by the language of the program that calls the SORT. program).
- Total length of all the sort keys cannot exceed 290 bytes or 580 4-bit "units."
  - Alphanumeric keys: Each byte in an alphanumeric key counts as one byte or two units. If the key is signed alpha (represented in COBOL by PIC S9 without COMPUTATIONAL), the high-order (or leftmost) unit of the key contains the sign.
  - Unsigned numeric keys: Every four bits of an UN field counts as one "unit."
  - Signed numeric keys: Every four bits of an SN field counts as one unit. The sign counts as an additional unit.

Output

The possible output media from the sorting operation include the following:

- Cards (coded in BCL or EBCDIC)
- Magnetic tape
- 100-byte disk
- 180-byte disk pack
- Line printer

Any records that are output to a line printer and that are smaller than a printer line (132 characters) appear left justified with space filled on the right. Any added content of output records that you define to be larger than the input records is comprised of all nulls. Output records that you specify as shorter than input records are truncated from the right.

Whenever a group of contiguous input records have equal key values, the order of those same records in the output file is unspecified.

Special Provisions for Sorting Variable-Length Records

The SORT. program can accept variable-length records on magnetic tape as input and can produce them for output. When you use variable-length records, the record size is stored, right-justified, in the first four bytes of each record. The maximum record and block sizes are specified in the FIB.

The MCP blocks the records into the output buffer when it writes a variable-length file until the buffer cannot hold another record. At that time, the MCP performs the physical writing and resumes blocking at the beginning of the buffer.

When the MCP passes the FIB for variable-length files to the SORT. program, it must treat each input record as a maximum length (FIBMRL) because the system does not permit variable-length records in random disk files. The system creates the work files for the SORT program sequentially and accesses them randomly.

A provision has been made for optimizing the way the SORT. intrinsic allocates disk space for variable-length records. In addition to the maximum record length, the SORT. intrinsic needs to know the record length that occurs most often. The parameter FIBVAR, which is the FIB through the four digits at FIB+106, contains this information.

FIBVAR must contain the length, in digits, for the most common record size in the file being sorted. FIBVAR must not be less than the size of the shortest record in the file and cannot exceed FIBMRL (maximum input record length). If you are unsure of the correct value for FIBVAR, you can use the value in FIBMRL. However, doing so disables the optimization logic of the disk allocation in the SORT. intrinsic.

Do not specify a sort key location that is outside the bounds of the shortest variable-length record you are going to sort.

If the SORT. program determines that any record input to it is not long enough to contain all the sort key fields specified in the parameters, the program deletes that record during the input (distribution) phase. A count of all such deleted records appears on the ODT before the program enters the merge phase. You can direct the SORT. program to continue with the truncated input file, or you can enter the DS command to discontinue the program.

Site-Specific Sort. Parameters

You can create SORT. parameter files to modify the sort parameters to do the following:

- Direct work files to multiple disk packs.
- Restrict work files to a single disk pack.
- Specify area assignments by file, by area, and by ID number.

The SORT. parameter files contain syntax that invokes the DMPALL utility program.

The SORT. parameter file named SORT.*s* where *s* is the system number establishes new SORT. parameters for that system. The SORT. parameter file named SORT.. (the word SORT followed by two periods) establishes new site-wide SORT. parameters.

These files consist of unblocked 80-character records. If either SORT.. or SORT.s is on disk at the time a user program calls the SORT. program, the SORT. program uses the file to modify the standard default parameter set. If both files are on disk, the SORT. program uses SORT.s.

The following is an example of a card file that invokes DMPALL to create SORT. parameter files.

```
?PFM CRDDSK PARAMS (SORT..)
?DATA PARAMS
$FILE
$WORK PACK
$PACK LESPAK
?END

?PFM CRDDSK PARAMS (SORT.2)
?DATA PARAMS.
$FILE
$WORK PACK
$PACK SYSTEM
?END
```

In this example, DMPALL sets site-wide parameters directing all sort work files to a disk pack named LESPAK. It then establishes parameters for system number 2 that direct work files to a disk pack named SYSTEM.

You can also create the control file using the CANDE Editor. Perform the following steps:

1. Make a file of type DATA.

2. Enter the parameter records.
3. Use the Editor COPY ALL TO DISK AS <name> command to convert the file to the format expected by the SORT. program.
4. Use the command ?CH <name> TO SORT.s or SORT.. to name the file according to what SORT. expects.

The parameters that you can set with a control file are $FILE, $AREA, $E.U., $WORK, and $PACK. Not all these options need to be set in each SORT. parameter file. If you include more than one value for a parameter, only the value that appears last is used. If the SORT. program finds errors in the SORT. parameter file when it executes, a warning message appears on the system ODT, and the SORT. program ignores the entire SORT. parameter file.

The following paragraphs describe each option.

$FILE

This record directs the SORT. program to place the entire file on one disk ID. To select the disk ID use a process based on the internal file number (FIBDFN in the file information block).

$AREA

This record-directs the SORT. program to place each area of the file on a specific disk ID. To select the disk ID use a process based on the area number.

$E.U.

This card directs the SORT. program to assign disks by disk ID number. The designation *EU* refers to the name of an entity analogous to the current disk ID. Enter one or two disk ID numbers after column 5. Disk IDs are numbered from 1 to 89. Each number must be two digits long, for example, 01.

- When one number appears, all sort work files are forced to the disk represented by the disk ID number.
- When two numbers appear, work files alternate between both disk IDs.

$WORK

This record directs the SORT. program to assign work files to disk pack. The reserved word PACK must appear after card column 5. User program parameters, which you enter through the SORT BCT, override this site default option.

$PACK

This record enables you to specify multifile pack IDs. This card must follow the WORK card. The first multifile pack ID begins after column 5 on the card, and you can specify up to two of them. The disk pack ID is from 1 to 6 characters long. If you use equal signs (=) in the name, the program recognizes them as masking characters and permits you to use any one of a number of disk packs. For example, if you enter a

disk pack ID LES===, then all disk packs with disk pack IDs beginning with LES are available to the SORT program.

Details on Site-Specific Parameters

Spaces function as word delimiters on all dollar sign ($) records.

You can disable the default override mode for the SORT. program at any time simply by removing the SORT../SORT.s parameter file or by changing it to a different name.

The maximum allowable buffer or single block size in any SORT. program file is 4998 bytes (characters). The SORT. program does not check this limit; if you exceed it, incorrect output could result.

The SORT work file Wnn000 is a very small file containing pointers that the SORT. program uses. If you use a SORT../SORT.s parameter file to place SORT. work files on disk pack or a specific disk ID, the Wnn000 work file is still assigned to system resource disk.

Error Messages

`ASCD/DSND ERROR SORT KEY <text>`

You have specified a sort key other than ascending or descending. The <text> indicates the relative key number and the error information. The SORT. program aborts.

`CLASS ERROR SORT KEY <text>`

A sort key contains an invalid class. The text includes the relative key number and the error information. The SORT. program aborts.

`FIBVAR SPEC ERROR; ALL RECDS SORTED AS MAXIMUM LTH.`

Input is variable length and the 4-digit parameter at FIB+106(FIBVAR) does not contain a valid size. The SORT. program continues.

`FILE <F-N> IS NOT A TRANSLATE FILE`

A check of the header before a translate table file is loaded reveals that the supplied file is not a valid translate file. The SORT. program aborts.

`ILLEGAL CONTENTS-RECORDS DISCARDED`

A key contains the value of][(hexadecimal 5A4A). The SORT. program continues.

`INSUFFICIENT MEMORY FOR SORT`

The SORT. program has tried to perform the requested job in the memory available but has found that it cannot proceed. The SORT. program aborts.

INSUFFICIENT MEMORY SORT OUTPUT

The output block length is too large to be processed in the amount of memory available. The SORT. program aborts.

<integer> VAR RCDS SHORTER THAN SORT KEY DISCARDED: ENTER YES TO CONTINUE SORTING: ELSE DS OR DP

During variable-length sorting, records were encountered that were shorter than the sort key and so were discarded. The SORT. program waits for you to enter YES, DS, or DP.

INVALID CLOSE CODE TO SORT

The CLOSE type requested for the input or output file is invalid. The SORT. program aborts.

INVALID INPUT FILE USERBLOCK DATA

User-blocking data has been specified for an input file, and this data conflicts with information in the file information block (FIB). The SORT. program aborts.

INVALID SIZE SORT KEY <text>

The size of a sort key is invalid. The text includes the relative key number and the error information. The SORT. program aborts.

INVALID LOCN SORT KEY <text>

A sort key has been specified beyond the end of the record. The text contains the relative key number and error information. The SORT. program aborts.

INVALID SORT INPUT HDW TYPE

Input must be from cards, tape, 100-byte disk, or disk pack. The SORT. program aborts.

INVALID SORT OUTPUT HDW TYPE

Output must be to cards, tape, 100-byte disk, disk pack, or printer. The SORT. program aborts.

INVALID OUTPUT FILE USERBLOCK DATA

User-blocking data has been specified for an output file, and this data conflicts with information in the file information block (FIB). The SORT. program aborts.

KEY FIELDS DELIMITER MISSING

The SORT. program is unable to identify the last sort key parameter. The SORT. program aborts.

```
PARITY ERROR BLOCK <integer> ; BLOCK USED
PARITY ERROR BLOCK <integer> ; BLOCK DROPPED
```

If a USE or PURGE on ERROR statement in the calling program causes the SORT. program to use or drop a parity error block, one of these messages appears on the ODT. The integer is the block number. The SORT. program continues.

```
PARITY USE OPTION INVALID
```

An invalid option has been passed for handling parity errors. The SORT. program aborts.

```
SORT INPUT VOLUME NOT NUMERIC
```

The number-of-records parameter contains hexadecimal values A-F. The SORT. program aborts.

```
SORT INPUT VOLUME NOT STATED
```

The number-of-records parameter is zero, and the SORT. program input is something other than 100-byte disk or disk pack. The SORT. program aborts.

```
SORT KEY <integer> OUTSIDE RECD; <text>
```

A key has been specified beyond the record size. The relative key number (<integer>) and information (<text>) appear on the ODT. The SORT. program aborts.

```
SORT RECORD COUNT ERROR: <integer-1> <integer-2>
```

The correct record count (<integer–1>) and the error count (<integer–2>) are displayed to indicate that the SORT. program realizes there is an error. If the same job repeatedly gives the same error count, there could be a hardware malfunction. Contact a Unisys Customer Service Engineering representative. The SORT. program aborts.

```
SORT TRANSLATE FILE <F-N> IS NOT ON DISK
```

A SORT using the translate option has been specified, but there is no file on 100-byte disk with the specified file name. Whenever the file is loaded, the sort proceeds. The SORT. program waits for you to load the file.

```
VARIABLE RECD SORT INTERNAL ERROR
```

The SORT program recognizes that it has made an error. If the same job repeatedly gives the same error message, contact a Unisys Customer Service Engineering representative. The SORT. program aborts.

```
VARIABLE RECORD FIB DATA SYNTAX ERROR
```

This message appears and the SORT. program aborts if you specified variable-length input and if neither input nor output are on magnetic tape, or if the user-blocking bit is set (on). The SORT. program aborts.

```
ZERO RECORD INPUT TO SORT INTRINSIC
```

This is a warning message only. The SORT. program continues.

Section 22
SORT:—Tape/Disk Sort Intrinsic Program

You can use the Tape/Disk Sort Intrinsic Program (SORT:) to sort one or more unordered files into a specified sequence. Specify tape, disk, or disk pack media as a sorting medium. The SORT: program is bound to the operating system. You can invoke it by using the SORT verb from within the program. The SORT: program is available for use only with the SRTUTL utility program and the COBOL ANSI-74 programming language. Refer to the *V Series COBOL ANSI-74 Compiler Programming Reference Manual (Volumes 1 and 2)* for more information.

Normal operating parameters for the SORT: program include work files on magnetic tape, 100-byte disk, or 180-byte disk pack. Work files on disk are assigned to default subsystems. Work files on disk pack are assigned on a space available basis to system resource (non-restricted) disk packs.

Memory Requirements

The memory limits for SORT: are the following:

- Minimum limit: 25K bytes
- Maximum limit: 499.5K bytes

Within the minimum and maximum limits, the operating system examines the following:

- The value of the LIMIT SRTMEM record in the system configuration file, discussed in the Volume 1.
- The memory occupied by the program that called the SORT: program.
- The memory value passed to the operating system in the parameters of the program call BCT. Some compilers make this value default to the size of the calling program.

The operating system allocates memory to the SORT: program based on the largest of these values. If the largest value is outside of the minimum or maximum limit, the operating system uses the limit value.

Pay particular attention to memory used for sorting when large record sizes are involved. Do not minimize sort memory too much when sorting large records. The number of tape or disk I/O operations and the number of sort passes depend on the number of records that can be evaluated in memory during the sort. As record sizes approach the maximum design value for a given program, the number of records that can be evaluated becomes relatively small if the memory available to the SORT: program is too minimal. For

example, if the record size is 4000 bytes or more, invoking the SORT: program with less than 100K bytes of memory causes a less than optimal sort performance.

Key Specifications

The SORT: program arranges the records of the input files according to specified keys within each record in the ascending or descending collating sequence that you specify.

All signed key fields are treated as computational items. If they appear in the record description as signed 8-bit, they will be packed into signed 4-bit before key comparison. All signed keys are sorted algebraically, and negative numbers are less than any positive number.

The sort key can consist of up to 40 separate fields, and the size of the combined keys cannot exceed 290 bytes, which are allocated as follows:

- One byte for each two digits of any computational key item.
- One byte for each two digits of any signed item plus one digit for the sign.
- One byte for each character of any other class of key item.

The first two in the previous list are rounded up to the next whole byte.

The SORT: program permits special COBOL trailing-sign bytes and separate leading-sign bytes. For these key types, the sign counts as a whole byte.

A single key field cannot exceed 98 units in length. In this context, units mean digits for computational items and bytes for all others.

Do not specify a sort key location that is outside the bounds of the records to be sorted. For variable-length records, this means the shortest record to be encountered throughout the run. If while reading the input file the SORT: program determines that any input record is not long enough to contain all the sort key fields specified by the calling program, that record is deleted during the input phase. A count of such deleted records appears on the ODT after the last input file has been read, and you can continue the sort without the offending records.

Input and Output Requirements

The SORT: program can accommodate input to and output from cards, pseudo card files, tape, disk, or disk pack medium that are supported by the operating system.

Input Restrictions

Any one input file must be wholly contained on one physical media type, but you can mix any media and physical block sizes in any combination. The record size (FIBMRL) must be identical for all files participating in the sort. For variable-length files, the maximum record length for the output file cannot be less than that for any of the input files.

The maximum record size (FIBMRL) for SORT: is 4998 bytes or 9996 digits. If the record size is larger than this, the SORT: intrinsic aborts.

The actual files you want to sort must agree with the declarations of those files as specified in the calling program. For example, if you declare an input file with one blocking factor and the actual file presented has some other block size, the SORT: program aborts.

When you specify multiple input files, you must still specify only one output file. If the calling program requires that more than 10 files be input to the SORT: program, the sorting memory requirement is increased by 500 bytes over the amount of memory that would otherwise be needed for each file after the tenth.

The largest block size you can use for SORT: depends on how much memory you can assign to the sort task. When you use large block size files, you increase the amount of memory required for efficient sorting. The largest block size for the SORT: program is 19998 bytes if you assign less than 150K bytes of memory to the sort task. If you assign 150K bytes or more, then the block size can be up to 29998 bytes.

Tape files consisting of variable-length records that are supported by the operating system can be processed by the SORT: program. However, if you use such variable-length records, you must specify all the files that participate in that sort as variable-length files. If you use variable-length records, the record size is stored right justified in the first four bytes of each record. The MCP blocks the records into the output buffer when writing a variable length file until the buffer cannot hold another record. When this situation occurs, the physical I/O is performed, and blocking resumes at the beginning of the buffer.

When the FIB for a variable-length file is passed to the SORT: program, treat each input record as the maximum (FIBMRL) length because variable-length records are not permitted in random disk files. The SORT: program creates work files sequentially and accesses them randomly. To optimize the allocation of scratch disk for variable-length records, the SORT: intrinsic requires the calling program to supply the length of the most common (that is, most frequently occurring) record in addition to the maximum record length.

The most common length is passed to the SORT: program in one of the interface parameters. When the SORT: program obtains this value, it breaks each variable-length record into fragments of that value. These smaller-sized fragments are processed

throughout the merge passes, which reduce both disk space required and time needed for the sort.

Input and Output Assumptions

Card, magnetic tape, disk, and disk pack input files must be in EBCDIC code. The SORT: program does not accept binary or ASCII input files. Blocked files on tape can consist of a variable number of fixed-length records (frequently the last block on a tape is a short block). The only exception is variable-length records on tape, where each record carries its own length in a 4-byte container at the front of the record.

Rules for Sequence of Output Files

The SORT: program has two rules for the sequence of output files. The program does not specify the relationship between the records in the output file within a given group of contiguous equal key records. Neither does it specify the ordering of those same records in any input file.

Tape Sorting

The Tape Sort mode of the SORT: program uses from three to eight work tapes in the sorting operation. You may specify the number of tapes in the interface parameters, but you cannot alter them at run time. If an input file is on tape or the output file is going to be on tape, that tape unit can serve its input function before being used as a work tape or its output function after having served as a work tape.

Work file units can be on any supported tape type; you can mix them in any combination. If you specify the sorted output as tape, that output will usually be directed to the tape unit that was first made available as one of the work tapes.

Tape Sort Operation

The SORT: program first acquires and then validates the control parameters from the caller. Next it opens the work file tapes. The number of work tapes that it needs is one fewer than the number you specified for the given sort.

Then the SORT: program opens input files, reads them one at a time, and writes them onto the work tapes in sequenced runs. When the program has read the last input file, it opens the last work file and performs a series of merge passes on the work files.

When the SORT: program determines that the number of runs remaining is less than the number of work file tapes, it opens the output file. Then the final merge operation writes the sorted file to the media that you specify.

Make sure that all scratch work tapes are large enough to hold the entire sort input because work files cannot be multireel files. If sort input is originally very close to being in sequence, most of the sort data can end up on one single work tape because work file

output reel swap during the input phase only occurs when a new sequenced run is to begin.

If sort data is more or less random, you can increase the total number of records that can be sorted in a single sort by increasing the number of tape units available to the program.

Disk or Disk Pack Sorting

Disk Sort Media

The DISK/PACK mode of SORT: uses work files on

- 180-byte disk pack
- 100-byte disk

Default Sort Media

The default sort media is 180-byte unrestricted disk pack. If the calling program so specifies, 100-byte default disk is used instead. The work file media is always on one type of media only, either all 100-byte or all 180-byte.

Work File Requirements

The amount of work file space that the SORT: program requires for a disk sort is approximately 2.5 times the size of the combined input files. If the work files you are using are on removable media, all volumes of the media must be present during the entire sorting operation.

Disk Sort Operation

The SORT: program acquires and then validates the control parameters from the caller. Next it apportions available memory according to the number of input records that are expected. When the SORT: program determines the number of input records, it opens a multiarea disk or disk pack work file large enough to hold the expected number of records.

Then the SORT: program opens the input files, reads one at a time, and writes sequenced runs to the work file. When the program has read the last input file, it reallocates the resources. This resource allocation is based on the number of records actually read and the number of merge passes that will be needed to merge the sequenced runs into a single run.

The purpose of this allocation is to minimize the number of passes to merge files consistent with the memory available to the program. Each pass reads a work file, which produces fewer runs of longer length, and then writes an output work file. These passes continue until the program observes that the next pass is to be the last. When the SORT: program reaches the final pass, it opens the output file and produces the sorted file.

Operating Considerations

The SORT: program does not use a specific method of work file allocation unless you are using a default-media override. If the work files end up on the same spindle or on disk units already experiencing a high level of activity, the performance of the SORT: program will be adversely affected. If work files are assigned to a disk ID that is also the prime disk ID containing the disk directory, then overall system performance may be adversely affected.

Default Media Override

No facility in the SORT/MERGE interface enables you to control the specific disk/disk pack storage units that should be used for sort work files. One reason for this is that such unit-specific information is frequently unknown when a program is compiled. Also the amount of space available for work files tends to vary significantly from spindle to spindle over time. For these reasons, assigning work files to specific media is deferred until run time.

Because of this assignment delay and because the MCP does not always know the particulars of your configuration or of its usage, the media to which the MCP assigns the work files is not always optimal. This fact can cause particular problems when the spindles with the most space available are the ones accessed most frequently.

Because of these potential problems, a media default override feature permits all decisions you make about sort intrinsic work files to be based on default media parameters that you supply to the system. (This feature is similar to the one implemented for the earlier SORT. program. The SORT: program uses a control file that affects only its operation. The syntax for this control file and for the control file used by the SORT. program are similar, but not interchangeable.)

The presence of a small control file on the system disk activates the media default system. This file is named *SORT*:s. The first five characters are always the same. The last character (*s* in this example) can be a digit 0, 1, 2, or 3, in which case the named control file applies only to the system represented by that number. If the last character is a colon, the control file applies to all systems. When both SORT:s and SORT:: are present, SORT:s takes precedence.

After you have created the control file, the program will do a syntax check and then reformat the control file. This process is performed only once, by the first sort that encounters a control file. All media names, subsystem numbers, and ID numbers specified in the control file should be mounted and ready at the time that the control file is first encountered by the SORT: program.

You can use the default media override feature to specify assignment environments for multiple sort work file media. You can use provisions in the SORT/MERGE interface and in the COBOL compiler to direct SORT: to use a specific environment number for a particular sort. You can change the pack name for any environment by modifying the control file. You can specify environment numbers from 1 to 90. Control file records not specified for a particular environment are for environment 0, which will be used with

programs that did not specify an environment number or programs that were compiled before the multiple environment feature was implemented in the COBOL compiler.

You can create the control file with the DMPALL utility program as follows, where *s* is the system number:

```
? PFM CRDDSK PARAMS (SORT:s)
? DATA PARAMS
? END.
```

You can also create the control file using the CANDE Editor. Perform the following steps:

1. Make a file of type DATA.
2. Enter the parameter records.
3. Use the Editor COPY ALL TO DISK AS <name> command to convert the file to the format expected by the SORT: program.
4. Use the command ?CH <name> TO SORT:x to name the file according to what SORT: expects.

The following paragraphs describe the various records that compose the control file. You need not include all types. The sequence of these records is not significant except where noted.

$SUPR

This record suppresses error messages that occur as a result of problems with the control file. Its specific function varies depending on its position in the control file.

- If $SUPR is the first record in the file, then after the first sort to sense and report an error this record suppresses all error messages that pertain to the default media override feature.
- If a $SUPR record appears in a non-zero environment, or if a $SUPR record appears in environment 0 but is not the first record, it affects only the environment it is in.

$ENVT

This record begins an environment declaration. All records that appear in the control file prior to the first $ENVT apply to environment 0. You must enter a two-digit environment number after column 5. All records from this $ENVT record to the next $ENVT record (or until the end of the file) apply to the environment that is being declared.

You do not have to declare environments in any particular order. If the media for a non-zero environment are missing when the control file is first processed, then the file will be initialized for the error-free environments only. When the SORT: program subsequently calls on the missing media environment, SORT: will display a warning message for each failed attempt to identify the missing media (unless all error messages have been suppressed with the $SUPR record).

$DFLT

This optional record is valid only for environment 0. It is possible that a program compiled to request a particular environment might be run on a system with a control file for which that environment number was not declared. When this happens, the default is to proceed with that sort as though the default media override feature were not being used (ie: no SORT:: file on the system). The $DFLT record overrides that default and forces environment 0 to be used in place of nonexistent environments if they are encountered.

$FILE

This record directs the SORT: program to place each work file on one disk ID (sets FIBDTK = 1 in the file information block). The disk ID is selected through a process based on the internal file number (FIBDFN in the file information block). You cannot specify $FILE when you use $AREA or $E.U.

$AREA

This record directs the SORT: program to place each area of the file on a specific disk ID (sets FIBDTK = 2 in the file information block). The disk ID is selected through a process based on the area number. You cannot specify $AREA when you use $FILE or $I.D.

$E.U.

This record assigns disks by disk ID number. The designation EU refers to a previous entity analogous to the current disk ID. You can enter one or two disk ID numbers after column 5. This record type exists to provide compatibility with SORT:: files prepared for use with earlier versions of this product.

$I.D.

This record assigns disks by disk ID number. You can enter one or two disk ID numbers after column 5. Disk IDs are numbered from 1 to 89. Each number must be two digits long and followed by at least one space (for example, 01 for disk ID number 1).

- When one number appears, all sort work files are forced to the disk represented by the disk ID number.
- When two numbers appear, work files alternate between both disk IDs.

$SUBS

This record assigns work files by disk subsystem number. You can enter one or two disk subsystem numbers after column 5. Each number must be a single digit followed by at least one space.

- When one number appears, all sort work files are forced to the disk subsystem represented by that number.
- When two numbers appear, work files alternate between both disk subsystems.

$PACK <family1> <family2>

This record assigns disk pack work files to family names 1 and 2. These disk packs can be restricted. If you specify only one name, all disk pack work files are directed to that disk pack family.

$WRKP

This record ignores the work file type specified in the calling program. The setting of the system option WRKP determines all work file assignments. Work files are assigned to 100-byte disk if WRKP is reset or to 180-byte disk pack if WRKP is set. You cannot specify $WRKP when you use $WORK. Refer to Volume 1 for a description of the USE WRKP system parameter.

$WORK DISK
$WORK PACK

This record ignores the work file type specified in the calling program. The media selection word DISK or PACK determines whether work files are assigned to 100-byte disk or 180-byte disk pack. You cannot specify $WORK when you use $WRKP.

The command word occurs in columns 1–5 of each record. The remainder of the record is free format. Spaces or commas are delimiters. The program checks the work media control file for errors and inconsistencies when you execute the SORT: program. If the program finds any errors, it ignores the file and proceeds without reformatting the file for use by future sorts. For example, the $FILE and $AREA records cannot both be present. Also, if you use the $I.D. record, you cannot use $FILE or $AREA.

The $WORK and $WRKP records are mutually exclusive. Such errors, or the inclusion of unidentified records, cause the following error message: <file name> MEDIA CONTROL FILE CONTAINS ERRORS. In addition, if you specify specific disk ID numbers or disk pack family names, those media should be present when you perform a sort. Otherwise, the message NON-PRESENT DPK/DSK/SUBS REQUESTED AS SORT WORK MEDIA appears on the ODT. Any syntax error makes the entire media override feature inoperative. Invoking a non-existent environment causes the following error message: ENVIRONMENT <number> IS NOT ACTIVE (MISSING MEDIA).

The default media for sorting is 180-byte disk pack.

Virtual Collating Sequence

This option alters the sequence in which the SORT: program arranges records during the sorting process. Normally, all characters that the program encounters in the sort keys are arranged in the hardware collating sequence of V Series Systems (that is, 00 through FF). Only those elements of the sort key described as unsigned alphanumeric are affected by the translation capability. Computational sort keys are always processed according to the hardware collating sequence.

The virtual collating sequence lets you do the following:

- Specify an entirely new collating sequence for the particular program invoking the SORT: program.
- Retain the normal collating sequence of 00 through FF, except for certain characters whose rank in the sequence you want to interchange.
- Make a number of characters have the same rank for the ordering of records.

The features of the virtual collating sequence option are particularly useful when you are using foreign alphabets or conversions from other computer systems.

Translate Option

You can invoke the SORT: program with the Translate option by specifying the name of the translate table file in the SORT CALL parameters. When you do so, the SORT: program acquires this file during initialization and uses it during all subsequent key processing. In this way, records are arranged as though the hardware collating sequence had been the virtual sequence that you specified.

Translate Tables

Any translate tables you want to use with the SORT: program (prepared by the MAKTRN program) must be on disk at the time you invoke the SORT: program. Refer to Section 13, "MAKTRN—Translation File Generator Program," for more information.

The names of the translate table files should be unique so that the files are not inadvertently sorted into the wrong sequence. The translate table file consists of a 400-byte single record, single-area file on disk. The SORT: program verifies the header information before it opens the Translate Table File to ensure that the file is a 400-byte single record, single-area file.

Error and Warning Messages

Each failure termination of the SORT: program displays an error message. For recoverable errors, continue the sort and the task that invoked the message in spite of the error. For all other errors, the SORT: program produces a memory dump and proceeds to error termination. Error termination cancels the caller and whatever job system the caller pertains to.

Non-Fatal Errors

The following non-fatal error messages can appear on the ODT.

`INPUT RECDS BEGIN WITH SORT MARKER CHAR`

This error message indicates that the SORT: program found input records that have the reserved character combination *// (5A4A)* in the first two characters. Input

records cannot violate this rule. Refer to "Recovery" later in this section for information about responding to this error.

RECORDS SHORTER THAN LAST POSITION OF SORT KEY

This error occurs when the SORT: program processes variable-length records and determines that at least one input record is shorter than some portion of the sort key. An example of this condition occurs when a sort key begins at byte 102 and an input file contains a 100-byte record. Refer to "Recovery" later in this section for information about correcting this error.

Recovery

During the input phase, the SORT: program discards all records that do not meet the requirements presented in this section. The program keeps a count of the number of records discarded. Appropriate error messages appear when the program has read all the input. You can continue the sort without the records that produced the errors. If you choose to continue, the program produces the sort file and returns control to the caller. If you choose not to continue, the SORT: program aborts.

Correctable Fatal Errors

The following paragraphs describe fatal errors that can occur. These error messages are self-explanatory and you can correct them. To recover, correct the error and restart the sort.

ALL FILES NOT DECLARED SAME RECD SZ

Except for variable-length sorts, all files included in the sort must have the same record size (FIBMRL). This error can also occur if the block size or record size for an input file does not exactly match the size declared for that file in the calling program.

KEY OUTSIDE RECORD OR SIZE ERROR

You have specified a sort key that either has zero length, is too long, or all or part of it is beyond the maximum record size FIBMRL.

MUST SPECIFY INPUT RECDS EXPECTED

You did not state explicitly the size of the file. File size cannot be determined from a header because input is not from disk or disk pack.

NUMBER OF TAPES FIELD INVALID

The number of tapes is not within the expected parameters of 3 to 8.

OUTPUT FILE IS DECLARED TOO SMALL

The capacity of the output file as declared cannot hold the number of records expected.

```
REC/BLK SIZE NOT AS DECLARED IN FIB
```

For disk and disk pack input files, the SORT: program checks the header when it opens a file to verify that the real record size equals the stated record size. This message indicates an inconsistency.

```
SORT KEY CLASS SPECIFIER INVALID
```

The program has found a key field that is not UN, UA, or any other permitted value.

Noncorrectable Fatal Errors

The fatal errors described in the following paragraphs are not normally correctable. The error can be an interface error possibly caused by the compiler that invoked the sort. The error can also be an internal SORT: program error.

If the specific error doesn't always occur, the hardware might be deficient.

If the specific error doesn't always occur and changes as you modify data, memory size, or sort keys, a program deficiency might be the cause. In such cases, you should save memory dumps and copies of the data and the calling program for analysis.

```
APPARENT MISSING TAPE BLOCK BCT 0794 FAIL OR PARAM SIZE ERROR
```

This error indicates a program CALL BCT failure. The parameters passed do not match MCP expectations.

```
DISTR PHASE MEM ALLOCATION ERROR
DUMMY RUN ASSIGNMENT FAILURE
DYNAMIC CODE RELOCATION FAILURE
DYNAMIC FIB RELOCATION FAILURE
INTERNAL SORT ASC/DESC FAILURE
MEMORY (DYNA) ALLOCATION ERROR.
MERGE PHASE MEM ALLOCATION ERROR
MERGE PHASE RUN CONTROL FAILURE
MERGE PHASE SEEK CONTROL ERROR
MERGE PHASE UNBLOCKING FAILURE
NOT ENOUGH MEMORY (PARAM READ)
NOT ENOUGH MEMORY (SORT BUFFERS)
OUTPUT PHASE MEM ALLOCATION ERROR
PASS TO PASS RECORD COUNT ERROR
```

These errors can result from internal inconsistencies in the SORT: program. Contact your Unisys Customer Service Engineering representative.

```
SMDP FIELD INVALID OP SPECIFIER
SMDP OP=01 MISSING OR DUPLICATE
SMDP OP=28,29 MISSING OR SEQ ERR
```

The SORT: program has either found no sort key specifiers or specifiers where they were not expected. This error can occur when a calling program, which is compiled

with a particular version of COBOL, is now running on a version of the MCP that invokes an incompatible version of the SORT: program.

```
SMDP OP=31,35,36 MISSING OR ERR. TRANSLATE FILE SPEC INV OR MISSING
```

The translate file name is too long, or the bracketing characters required around it in the parameters are missing.

```
VARIABLE LENGTH PARAMS DO NOT AGREE
```

The SORT: program has found a variable-length file that was not expected. If variable-length sort is active, all files participating in that sort must be specified as variable-length files. (FIBBLK = 2).

```
VAR RECD INPUT PROCESS ERROR VAR RECD MERGE PROCESS ERROR VAR RECD OUTPUT
PROCESS ERROR
```

The SORT: program has detected a failure in the management of variable records and portions of records in unexpected locations.

Section 23
SRTUTL—Generalized Sort Utility Program

Overview

The Generalized Sort Utility program (SRTUTL) sorts sequential, relative, or indexed sequential files by keys. It then generates either an ADDROUT file acceptable to RPG or a new sequential file of complete records. Both RPG and COBOL users can use the SRTUTL program. However, many functions that the SRTUTL program provides are available within the COBOL compiler.

Indexed and relative files that you sort with the SRTUTL program must have been created by a program compiled with a compatible release of COBOL or RPG. Additionally, you can selectively designate or drop input records before you actually begin sorting.

Syntax

The SRTUTL program accepts both a primary and secondary source file, each of which defines the sort requirements and then generates a sorted file in the format you request.

The SRTUTL program ignores anything that you enter following a percent sign (%). The percent sign indicates that the data comprise a comment. These comments appear on the output listing.

The SRTUTL program does not have an Editor file type reserved. If you plan to maintain the SRTUTL program parameters in Editor files, you should use file types whose sequence numbers are in positions 73 to 80 (ADS, BPL, FORTRAN, DASDL, WFL).

If you use the DASDL format, you can include the SRTUTL program jobs in SYS COMP because the compile commands are identical.

Program statements for SRTUTL can occur in any order as input to the SRTUTL program. You enter the program statements in free format into positions 1 to 72 of an input record. Positions 73 to 80 are reserved for sequence numbers. The input to SRTUTL consists of the series of program statements that appear in Figure 23–1.

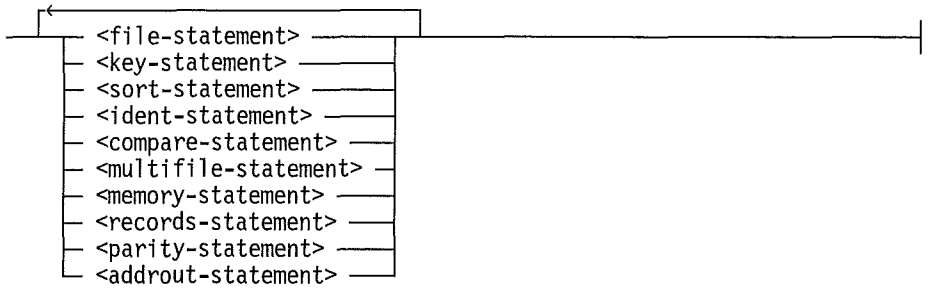


Figure 23–1. SRTUTL Program Statements

FILE Statement

The file statement enables you to describe the format both for the input file you want to sort and for the output file that results from the sort. Be as accurate as possible when you enter the descriptions, especially when you are describing indexed and relative files because these file types store information in a control block in the maximum record position of the file.

All FILE statements must include the medium, the record length, and the blocking factor for each file. Also specify in the statement either the file identifiers, the close with PURGE disposition of the input file, or both.

You must begin the FILE statement with the reserved word FILE followed by the reserved word IN and an input file description; next enter the reserved word OUT, and an output file description.

The syntax for the FILE statement appears in Figure 23–2.

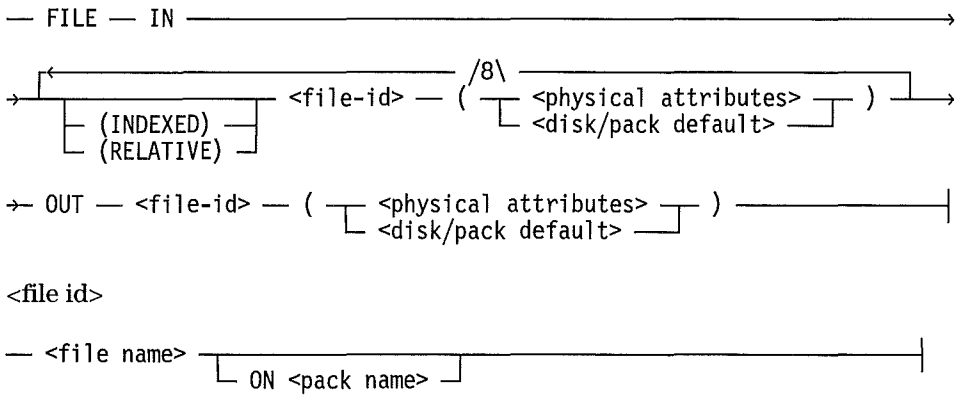
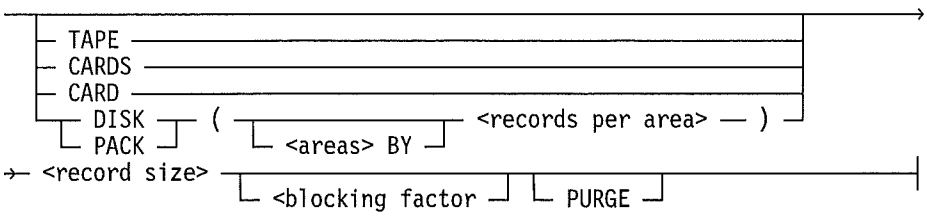


Figure 23–2. FILE Statement

<physical attributes>



<disk/pack default>

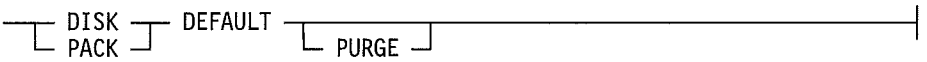


Figure 23–2. FILE Statement (cont.)

The following paragraphs describe the components of the FILE statement.

FILE

This keyword begins each <file statement>.

IN

This keyword begins the input file characteristics.

OUT

This keyword begins the output file characteristics.

(INDEXED)
(RELATIVE)

These options identify the input file as an indexed or relative file. If you do not specify either, , then the program assumes that the file is a normal sequential file.

<file id>

Be sure that this ID label conforms to the standard format of 1 to 6 alphanumeric characters, the first of which must be alphabetic. If the file identifier includes any special characters or if the identifier is a reserved word, you must enclose it in quotation marks. If you specify more than one <file id> in the FILE IN phrase, then multiple input files will be merged into a single output file. If you want more than one file to be sorted without being merged, use the MULTIFILE statement instead of multiple input file names in the FILE IN phrase.

<physical attributes>

Use this option to define the hardware on which the input and output files reside. Enter one of the following reserved words in the definition:

TAPE —Tape file

CARD or CARDS—Card file

DISK—Disk file

PACK—Disk pack file

<disk/pack default>

You can invoke default settings for both input and output files.

- Input file: If the input file is on disk or pack, you can specify DEFAULT. The number of areas, records per area, record-size, and blocking factor is automatically obtained from the disk file header. If this information is not available, you must specify the number of records for each area. The default number for the number of areas is set to 100.
- Output file: The default value for records per block for the output file is set to 1. The default value for number of areas is set to 20, and the default value for records per area is set to 2000. The individual record length is equal to the record length that you have established for the input file.

<areas>

If you do not use the DEFAULT option, you must specify the number of records per area for all output files and for any input file on disk or disk pack. Also, you can elect to specify the number of areas; if you do, you must enter the word BY following the number. The default value for number of areas is 100.

<record size>

If you do not use the DEFAULT option, you can use the record size option to specify the logical record length in characters. This specification is required. Input records may not be of variable length. The maximum record length accepted by the SRTUTL program is 4,998 characters. If the record size you enter is not divisible by 2 (that is, an even number of words), the SRTUTL program pads the size.

<blocking factor>

Enter the block size to define the number of logical records in a block. This value cannot exceed 999 per block. If you omit a blocking factor, the SRTUTL program assumes fixed-length records, blocked 1.

PURGE

You can specify the disposition of the input file as close with PURGE. Indicate this disposition by entering the word PURGE. If you omit this option, the object program closes the input file with RELEASE.

You must enclose in parentheses and delimit by a space all options that indicate media, record length, block size, and disposition.

If you declare more than one input file but do not declare a SORT statement, then the SRTUTL program merges the two files.

Example

```
FILE IN(INDEXED) "DISK" (DISK(20) 200 10)
OUT SORTED (PACK (20 BY 100) 200 5)
```

In this example, the FILE statement describes an indexed sequential input file called DISK. The file resides on disk, has 20 records per area, and is composed of 200-character records blocked 10. The output file is to be written to disk pack (20 areas, 100 records for each area) and is to be composed of 200-character records blocked 5. The input file ID DISK is enclosed in quotes because DISK is a reserved word in the SRTUTL program.

MULTIFILE Statement

The MULTIFILE statement allows you to name additional input files to be included in the sort. The <file-id>s included in the MULTIFILE statement are in addition to the <file-id> included in the FILE IN phrase. Each of the files listed in the MULTIFILE statement is sorted, but the files are not merged together. If you want the sorted files to be merged into a single output file, use multiple <file-id>s in the FILE IN phrase.

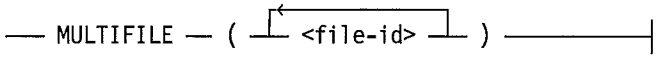


Figure 23–3. MULTIFILE Statement

KEY Statement

The KEY statement enables you to describe the keys by which the file should be sorted. You must begin the key description with one of the reserved words: KEY, KEYS, FIELD, or FIELDS. The reserved word must be followed by a description of each key.

The syntax for the KEY statement appears in Figure 23–4.

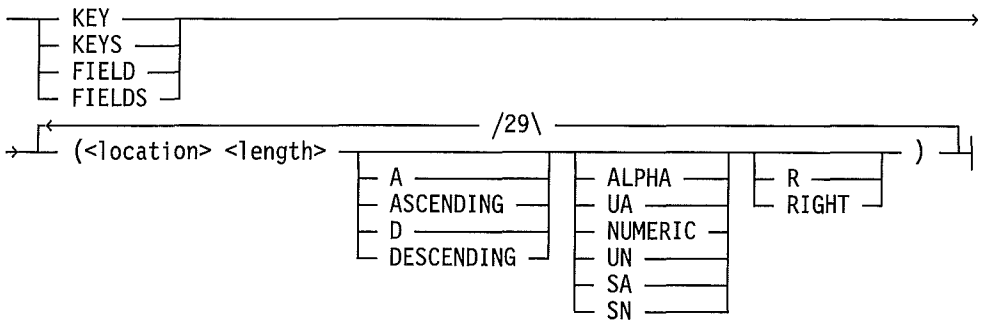


Figure 23–4. KEY Statement

The following paragraphs describe the components of the KEY statement.

location

The location is the most significant character or digit of the sort key related to the beginning of the record. The first character or digit of the record, depending on the data type you are using, is considered to be location 1. If the field you want to sort is signed numeric, the location is the sign digit. For a sort on signed alphanumeric data, the location is the character containing the sign.

length

Use this option to specify the number of characters or digits in the sort key, depending on the DATA TYPE. You must include this parameter. The sign character is included in the length of a signed alphanumeric field, but the sign digit is not included in the length of a signed numeric field.

Sequence

Use this option to specify the sequence to which the field should be sorted. The SRTUTL codes an A or ASCENDING for ascending sequence and a D or DESCENDING for a descending sequence. If you omit the code, the SRTUTL program assumes an ascending sequence. You can mix ascending and descending sequence sort keys within the total key.

Data Type

You can specify the type of data to be sorted by entering one of the following reserved words:

- ALPHA OR UA for unsigned alphanumeric (data is in bytes)
- NUMERIC OR UN for unsigned numeric data (data is in digits)
- SA for signed alphanumeric data (the sign is in the most significant digit of the most significant character and the data is in bytes)
- SN for signed numeric data (data is in digits)

Sign Position

If you do not enter an R or RIGHT here, the data will be defined as left-signed SA or SN. For right-signed SA data, the sign will be in the most significant digit of the rightmost character. For right-signed SN data, the sign will be in the rightmost digit. You cannot enter a UN or an ALPHA data type in this position.

You can specify different data types for individual keys within a total key. If you omit the data type, the SRTUTL program assumes an unsigned alphanumeric.

Enclose each field description in parentheses and separate it from other descriptions with a space. You can delimit the parameters within an individual description by a space.

The individual number of sort keys that the SRTUTL program permits is 30. The total key must not exceed 100 units. The total key is the field that results from the concatenation of the individual keys. The units are bytes (characters) for UA and SA keys whereas the units are digits for UN and SN keys. The combined length of all keys, including any padding digits necessary to align keys to byte boundaries, must not exceed 560 digits.

When you enter fields that overlap, they are not acceptable. For example, the specification KEY (3 5) (4 6) will be flagged as a syntax error.

Note: *A key containing the character combination]] (the characters right bracket-left bracket, the EBCDIC representation of hexadecimal 5A4A) is illegal.*

Example

```
KEY (5 5) (25 2 SN) (30 3 D)
```

In this example, the KEY statement causes a sort that proceeds in the following manner:

- First, by major key, which is five unsigned alphanumeric characters beginning in the fifth character position of each record, into ascending sequence
- Then by two signed numeric digits whose sign is located in the twenty-fifth digit position of each record, into ascending sequence
- Finally by the minor key, three unsigned alphanumeric characters, beginning in the thirtieth character position of each record, into descending sequence.

SORT Statement

The SORT statement enables you to describe the type of sort you want performed. If you omit this statement, then the SRTUTL program assumes the HOTSORT option unless you declare more than one input file in the FILE statement. In that case, the SRTUTL program performs a merge operation.

You can define up to a maximum of eight files as input to merge. The SRTUTL program assumes that each of the input files you define are already in the desired sequence.

The syntax for the SORT statement appears in Figure 23–5.

Figure 23–5. SORT Statement

HOTSORT

Use this option to perform the following functions:

- Read a relative, indexed, or sequential data file sequentially.
- Call the sort utility SORT: to sort the records, using disk for the work files.

TAPESORT

Use this option to designate the number of tapes that will be used for the sort. The number of tapes you enter must be between 3 and 8.

DISKSORT

Use this option to read a relative, indexed, or sequential data file, and call the SORT. program to sort the records, using disk for the work files.

INCLUDE
DELETE

If you omit these options, the SRTUTL program calls the sort directly. Otherwise, it creates an intermediate disk file with the same name as that of the output and displays the number of records to be sorted.

ADDROUT Statement

You can use the ADDROUT statement to automatically select the requested sort keys (refer to the KEY statement in this section), concatenate them, and then write a file of records. This file should consist of an 8-digit relative record number, plus the concatenated keys. ADDROUT then uses the concatenated keys to request a sort. The default value for the sort is HOTSORT.

The syntax for the ADDROUT statement is shown in Figure 23–6.

Figure 23–6. ADDROUT Statement

IDENT Statement

The IDENT statement enables you to assign a specific name to the SRTUTL output listing. If you omit this statement, spaces instead of a program name appear at the top of the listing instead of a program name.

The syntax for the IDENT statement appears in Figure 23–7.

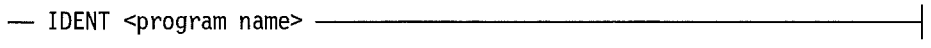


Figure 23–7. IDENT Statement

The program name must be from 1 to 6 alphanumeric characters long, the first of which must be alphabetic. If you use special characters in the program name or include reserved words, you must enclose them in quotation marks.

COMPARE Statement

Use the COMPARE statement to specify the inclusion or exclusion of records.

The syntax for the COMPARE statement appears in Figure 23–8.

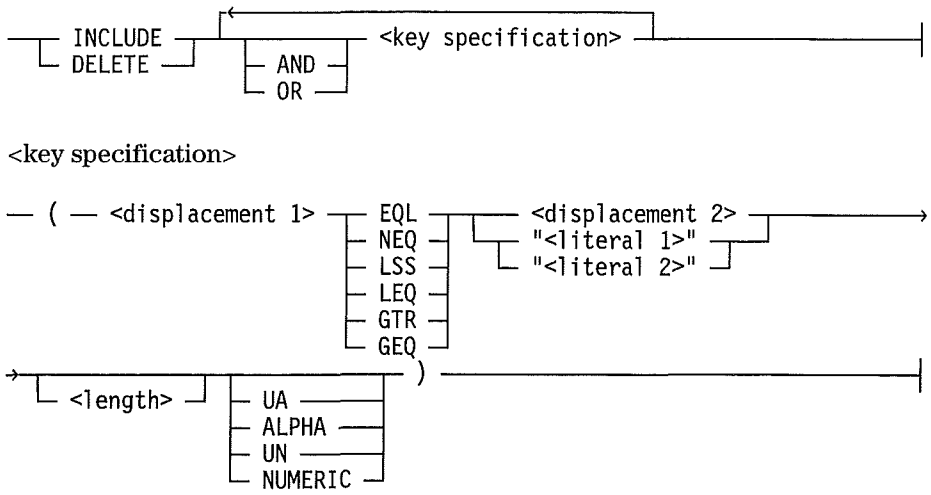


Figure 23–8. COMPARE Statement

The words INCLUDE and DELETE are reserved words. You can enter them only once within a set of sort specifications. However, one or more keys can follow each of these words. If you specify both INCLUDE and DELETE, the program applies INCLUDE first.

You can join multiple keys within INCLUDE or DELETE by using the reserved words AND or OR. However, the total number of keys you specify cannot exceed 10. The resulting expressions are then evaluated element by element, left to right.

Caution

If OR is encountered and all previous expression elements are satisfied, the evaluation is terminated as TRUE. If AND is encountered but not all previous expression elements are satisfied, the evaluation is terminated as FALSE. In all other circumstances, the entire expression is evaluated to a TRUE or FALSE conclusion.

You can use the first operand (displacement 1) to specify the offset of the key from the beginning of the record. If you specify ALPHA as the key, the offset is in bytes. If you specify NUMERIC as the key, the offset is in digits.

You can specify the length of the field in the length field. If you omit this specification, the default value is 1. The entry that follows the key displacement is the relational operator

that is used in comparing the specified operands. Valid entries are EQL (equal), NEQ (not equal), LSS (less), LEQ (less or equal), GTR (greater), and GEQ (greater or equal).

You can use the second operand (displacement 2) to specify the displacement into the record or as an alphanumeric or hexadecimal literal. As a displacement, the second operand is subject to the rule outlined for the first operand.

Hexadecimal literals are delimited by @ (the "at" sign), and can contain 1 to 6 hexadecimal digits. Character literals are delimited by " (quotation marks) and can contain up to three characters. All characters are valid with the exception of the quotation or question mark. Literals are padded or truncated as necessary, depending on the setting of the length attribute.

Use the length field to specify the length of the operands to be compared. If you omit this specification, the default value is 1. Use ALPHA and NUMERIC to specify whether the operands are character or digit oriented. You can abbreviate these entries to UA and UN, respectively. If you omit this specification, the default is UA (ALPHA).

MEMORY Statement

Use the MEMORY statement to increase the memory that the SORT: program is to use (up to 999 KD). The default for this setting is digits. Normally, the SORT: program uses the same amount of memory as the SRTUTL program.

The syntax for the MEMORY statement appears in Figure 23–9.

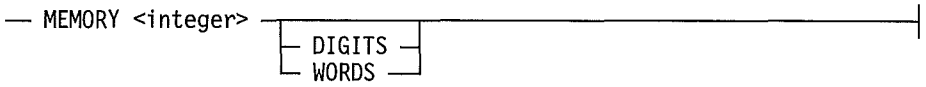


Figure 23–9. MEMORY Statement

PARITY Statement

The PARITY statement enables you to drop all unreadable records that are identified during the actual sort. The default value for this statement is to not have the option set. In this case, any parity errors will cause the program to stop.

The syntax for the PARITY statement appears in Figure 23–10.

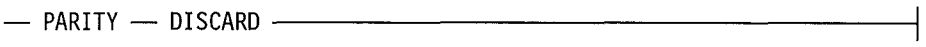


Figure 23–10. PARITY Statement

RECORDS Statement

The RECORDS statement enables you to optimize the sort operation by estimating the total number of records to be sorted. If you also enter INCLUDE and DELETE statements, the SRTUTL program calculates this number. Otherwise, the SORT: program attempts to determine file sizes from disk or disk pack headers. If the input is from tape and the sorting medium is not tape, you must enter the RECORDS statement.

The syntax for the RECORDS statement appears in Figure 23–11.

```
— RECORDS <integer> ——————————————————————————————————————————|
```

Figure 23–11. RECORDS Statement

Executing SRTUTL

You can execute the SRTUTL program in one of the following ways:

- With a card deck (refer to Figure 23–12)
- With an INSERT statement (refer to Figure 23–13)

The SRTUTL program, by default, expects a card file as input. You can override this default value by adding the INSERT clause to the EXECUTE command.

```
    ⎧ EXECUTE                   ⎫
    ⎪ EX                        ⎪
  ? ⎨ COMPILE user-id [WITH]    ⎬  SRTUTL
    ⎪ CMP                       ⎪
    ⎩                           ⎭

  ? DATA CARD
     program statements
  ?END
```

Figure 23–12. SRTUTL Execution Card Deck

```
    ⎧ EXECUTE                   ⎫
    ⎪ EX                        ⎪
  ? ⎨ COMPILE user-id [WITH]    ⎬  SRTUTL IN 0 4 UA "abcd"
    ⎪ CMP                       ⎪
    ⎩                           ⎭
```

Figure 23–13. SRTUTL Execution without Cards

Execution Commands (EXECUTE, COMPILE)

You can initiate the SRTUTL program with either the EXECUTE or the COMPILE command. The advantage of using the COMPILE command is that additional features are available:

- In ODT displays, system logs, and Work Flow Language (WFL) job summaries, the program name is represented as "<multi-program identifier>/SRTUTL." The multi-program identifier can be any 1 to 6 character identifier, such as the name of the file being sorted or the name of a program in a logical job stream. Thus, you can easily identify individual copies of the SRTUTL program in a multiprogramming mix.
- If the SRTUTL program detects errors in the input parameter statements, it notifies you because the execution of subsequent programs can be affected. The message ** SYNTAX ERR appearing on the EOJ line in the ODT display or on the EOT line in a WFL job summary indicates an error. The SRTUTL program removes programs linked to this execution by the MCP command AFTER from the schedule if you set the MCP AFTR option and if SRTUTL detects an input parameter error. Uniqueness for job streams is possible by using the multi-program identifier.

Example

```
?COMPILE PA110 SRTUTL; FILE CARD PA110C
?EX PA120 AFTER PA110/SRTUTL
```

In this example, the SRTUTL program executes PA120 if you set the MCP option AFTR, and only if it finds no errors in the parameters and goes to a normal end-of-job.

Executing SRTUTL through WFL

A WFL job can test for successful processing of the input parameters with the COMPILEDOK syntax, as shown in the following example.

Example

```
COMPILE PROG1 WITH SRTUTL [P1]; IF P1 IS COMPILEDOK THEN...
```

In this example, if you initiate the SRTUTL program with EXECUTE (or, in WFL, RUN), errors in the input parameters are noted only in the output listing, and the SRTUTL program goes to a normal end-of-job. The program does not notify you of the errors, and subsequent programs assume that the sort has occurred.

SRTUTL Memory Usage

The SRTUTL program allocates dynamic memory to hold buffers, and includes and deletes code. Normally, the space is adequate. However, if the files to, and other output from, the sort have very large blocks, you might need to enter a ?MEM =nn statement.

Cardless Executions

In the INSERT clause, a, b, c, and d represent alphabetic character positions. Fill each position with a letter or a blank. Enclose strings in quotation marks. Always leave position c blank.

Each position within the INSERT clause represents an input file or file format. The character in the position indicates either the device type on which the file is located or the format of the file (refer to Tables 23–1 and 23–2).

Table 23–1. Valid INSERT Values for Cardless Execution

Position	Represents	Permissible	Default
a	Primary Symbolic File	C, D, P, T, s	C
b	Secondary Symbolic File	C, D, P, T, s	D
c	Not Used	s	s
d	Primary File is Editor Format	E, s	s

s = space

Table 23–2. Meanings of INSERT Values for Cardless Execution

Position	Character	Meaning
a, b	C	Card Reader
	D	Disk (any 100-byte)
	P	Disk pack (180-byte)
	T	Magnetic Tape
	s	device unspecified
d	E	Editor-format primary symbolic file
	s	Primary symbolic is non-Editor file

s = space

You can indicate the input files by specifying the appropriate combination of permissible values from Table 23–2 in positions a, b, and d of the INSERT clause. After you have specified the media and file format, you can assign file names on dollar records or in file equate clauses if the actual names of the files are not the default names that the SRTUTL program assumes.

Table 23–3 summarizes the input and output files and the formats that are acceptable when you use SRTUTL (an X indicates an acceptable file). It also summarizes SRTUTL's internal name for each file (for use in file equating) and the values that must be supplied in the INSERT clause when you execute without cards.

Table 23–3. SRTUTL Files and Format

Input/ Output File	File name	80x1 Card file	80x Disk file	EDIT file on disk	80x Disk pack File	Disk- pack edit file	80x Tape file	132x1 Print file
Primary symbolic	CARD or EDITFL	X	X	X	X	X	X	
Insert value		(a=C)	(a=D)	(a=D, d=E)	(a=P)	(a=P, d=E)	(a=T)	
Secondary symbolic	SOURCE	X	X		X	X		
Insert Value		(b=C)	(b=D)		(b=P)		(b=T)	
Output Listing	PRINT						X	

Note: *Source input files can use any blocking factor from 1 through 9. The name of the primary file is CARD when it is a non-Editor format file, and EDITFL when it is an Editor file.*

DISK is the default medium for the primary symbolic file when you specify Editor format.

The values you specify in an INSERT clause take precedence over information in dollar records. For instance, a dollar record can supplement the information in an INSERT statement by providing the name of the secondary input file.

Examples

```
EXECUTE SRTUTL IN 0 1 UA "D" FILE CARD = PROG1S
```

In this example, input specifications for the SRTUTL program are in a disk file named PROG1S. There is no other input file.

```
EXECUTE SRTUTL IN 0 4 UA "DP E" FILE EDITFL = PATCjc
```

In this example, there are two input files containing the input specifications for the SRTUTL program. The primary symbolic file is an Editor file on disk, named PATCjc. It contains records to add to other records or to replace other records in the secondary file. The secondary symbolic file is on disk pack; its name and the name of its disk pack are specified on a dollar record (not shown).

Label Equation

Label equation cards must be used to uniquely identify source program files in a multiprogramming environment. The internal file names and external file identifiers for the SRTUTL program files appear in Table 23–4.

Table 23–4. SRTUTL File Names

Internal File Name	External File ID	Function
CARD	CARD	Non-Editor primary symbolic file
EDITFL	EDITFL	Editor Primary symbolic file
SOURCE	SOURCE	Secondary symbolic file
PRINT	PRINT	Printed output listing
PDATAB	PmxTpB	SRTUTL work file, where mx indicates mix- number and p is the system-number

When you want to change the external file identifier on a primary source language file, the label equation statement you would use is the following:

```
?FILE CARD = <file ID>
```

The file ID is a unique file identifier of six characters or fewer.

You can also direct files to specific hardware devices with the MCP FILE statement. For example, to direct the SRTUTL program work file to a specific disk pack, you can use the following label equation:

```
?FILE PDATAB = pack ID/PDATAB DPK
```

The pack ID is the name of the disk pack you want.

Note that by default the SRTUTL work file resides on the same hardware device as the output sorted file.

You can specify punched cards, disk, disk pack, or magnetic tape as a source language input medium for a single input file or for a master file. If you use two input files, SRTUTL merges them according to sequence number.

Records of input files are 80 bytes long. You can use any blocking factor between 1 and 9 inclusively for input from disk, disk pack, or magnetic tape.

Dollar Options

Certain options are available to you during operation, and you can activate or deactivate them with "dollar" options. These options have a dollar sign character ($) in column 1 and can be interspersed at any point in or immediately before the primary or secondary input file. You can specify one or more dollar options following the dollar sign character ($).

If you do not have control records, the default entry is LIST, and a single card deck is expected as input.

The dollar options SET and RESET enable you to turn on or turn off options you specify on a given line without affecting any other options that you do not specify. When SET or RESET is not the first dollar sign ($) option listed on the dollar sign ($) card, the options listed are turned on and all others are turned off.

The following paragraphs describe the dollar options available to you.

CHECK

This option flags sequence errors with a warning message.

CLEAR

This option means that all options that do not have associated values are reset, except MERGE.

CORRECTOK

When you set this option in a dollar record, the SRTUTL program attempts to correct certain syntax errors in the sort parameters. When the SRTUTL program detects an error, that it can attempt to correct, the error is flagged with a message in the following form:

```
CORRECTED: <text of error message>
```

The SRTUTL program prints the word CORRECTED when it detects a correctable error, regardless of whether it actually makes the correction. However, the SRTUTL program does not actually make the correction unless you have specified $CORRECTOK.

DELETE

This option appears only in the primary source file, and the program ignores it unless you set $MERGE. Images from the secondary source file are discarded (including other dollar options) until you reset this option. Items discarded are not listed in the execution listing unless you set the LISTDELETED option.

DOUBLE

This option produces a double-spaced listing of source images processed and of any resulting error messages.

ERRORLIMIT = nnn

With this option, execution is terminated when the number of errors you specified in nnn occurs, where nnn represents a 3–digit integer with a default value of 100.

LIST or LIST$

This option produces a single-spaced listing of source images processed. The program uses this option as the default value unless you reset it.

LISTCCI

This option produces lists of all dollar options in the output listing. Another name for dollar options is compiler control images (CCI).

LISTDELETED

This option lists deleted or voided images in the output.

LISTOMITTED

This option lists omitted images in the output.

LISTP

This option lists card images only (not source file) when you reset LIST.

MERGE ["file identifier"] [device]

This option permits you to compile a program with two symbolic input files. The inputs are merged on the basis of sequence numbers, one input being the primary input file that contains this option. The file identifier is either file ID or multifile ID/file ID, where file ID and multifile ID must be alphanumeric literals, each not more than six characters long. The file ID specifies the file name of the secondary input file. SOURCE is the assumed file name if none is specified; however, you can override this file ID with label equation. Device is the hardware on which the second input file resides. Valid devices are DISK, TAPE, and disk pack. DISK is the default.

Once you set this option, you cannot reset it and it remains in effect for the entire program.

OMIT

This option appears in both primary and secondary source input files. Images from any source are ignored (but passed to new source) until you reset this option. Items omitted are not included in the output listing unless you set the LISTOMITTED option. Any dollar options encountered while OMIT is set are processed normally.

PAGESIZE = nn

This option specifies that the maximum number of lines for each page of the output listing is equal to nn, where nn represents a 2-digit integer with a default value of 56.

SEQ nnnnnn +nnnnnn

With this option, source images are resequenced starting with the sequence number nnnnnn and incrementing by +nnnnnn. The default start (or base) is 10, and the increment defaults to +10.

SEQCHECK

Refer to the CHECK option in this section.

SEQUENCE

Refer to the SEQ option in this section.

SUMMARY

This option produces a listing of the execution summary when you reset the LIST option. This summary is included in the LIST option.

VOID nnnnnn

This option discards images from secondary source files until a sequence number larger than nnnnnn is encountered in the secondary source file.

Section 24
SYSTEM/COPY—File Transfer Utility Program

Overview

The SYSTEM/COPY program is a bound intrinsic that performs file transfer and directory functions for the V Series MCP. You invoke the SYSTEM/COPY program with the commands COPY, MOVE, COPYADD, COMPARE, MERGE, and DIR. Refer to Volume 2 for more information.

The SYSTEM/COPY program performs the following functions:

- Copies files from one or more sources at a time
- Merges files from one or more source tapes and from disk and disk pack to one or more tapes
- Compares data between source and destination files
- Copies files to one or more destinations at a time
- Changes the names of files as they are being transferred
- Copies files from tape to their original volumes
- Permits flexibility in job scheduling and AFTER executions because simultaneous executions of COPY can use a unique multiprogram ID
- Stores files in requested sequence on all output tape volumes
- Produces two tape formats: ICTAPE (interchange format) and TAPE
- Recognizes and reads automatically a variety of tape formats: LOADMP (type 5), PACKUP (type 6), TAPE, and ICTAPE
- Permits the use of large, multilevel file names on tapes
- Copies from any reel of a multireel tape set because each reel includes a directory of the files on it and all subsequent reels
- Supports 38000 BPI magnetic cartridge tapes (MTC)
- Supports 9-track tapes (NRZ, PE and GCR)
- Lists directories for disk, disk pack, and tape. Refer to the DIR command in Volume 2 for more information

Program Initiation

You can initiate SYSTEM/COPY in either of two ways:

- You can use any of the commands COPY, COMPARE, MOVE, COPYADD, or DIR to present SYSTEM/COPY syntax to the operating system. The command syntax is given in the *V Series Systems Operations Guide Volume 2: System Commands*. The operating system initiates the SYSTEM/COPY program, places the syntax in a pass file and submits the pass file to the program. A pass file is a disk file that contains the command syntax and that is used to pass the syntax to SYSTEM/COPY.
- You or an application can prepare the pass file. Then you execute the SYSTEM/COPY program, giving it the name of the pass file. For applications that generate large SYSTEM/COPY requests via pseudo card reader decks or programmatic ZIP statements, this option can improve performance by reducing reparsing of the syntax.

  The pass file is expected to be a disk file with 100-byte records, blocked one record per block. The file name is as follows, where *mm* is a unique pass file number and *s* is the system number:

  %*mm*0*s**

  To use this option, initiate SYSTEM/COPY with an INSERT statement with the unique file pass number at address 0 for a length of 2 digits, as follows where *mm* is the pass file number:

  EXECUTE COPY; INSERT 0 2 *mm*

  SYSTEM/COPY purges the pass file after using it.

Program Flow

This section provides a high-level program flow description for the SYSTEM/COPY program. The operating system (MCP/VS) executes the SYSTEM/COPY program directly and receives the required syntax from a pass file.

When it is initiated, SYSTEM/COPY receives a pass file as described earlier in "Program Initiation." The program first analyzes the syntax received from the pass file. Warnings or error messages for problems that the program encounters are displayed. If the program does not find fatal errors it acquires memory to accommodate the necessary number of file structures and buffers, based on the values of the BUFFERS and BUFFERSIZE options in the COPY command syntax and the COPYBUFS and COPYBUFSZ limits in the system configuration file. If there is insufficient memory to accommodate the memory size requested, the program might be marked as PR STOPPED (priority stopped).

When the program has acquired the appropriate memory, file expansion occurs. For those source volumes that require it, group specifiers are then expanded to their respective single-file names. The program also uses tape-drive space efficiently in the following way:

- If there are any destination-tape volumes, the program searches tape-source volumes for both group specifiers and single files.
- If destination-tape volumes are not involved, source-tape file expansion is postponed until that volume is actually needed for transfer. This technique provides better use of tape drives.

When the program has generated a complete file list for all sources and a tape directory for any destination tape volumes, the SYSTEM/COPY program is ready to perform the required tasks.

Beginning with the first source file, the program uses the file list associated with that source to perform the required functions. This process continues for each specified source volume and its associated file list.

Each file transfer proceeds when the program finds and opens the file on the current source volume. If the COPYADD function is in progress, each destination volume is checked for the presence of the destination file name, which may differ from the source file- name when you use the AS capability. Then the program opens all destination files, and each file is transferred with minimum I/Os and with effective use of buffers. As a general rule, mixing destination types (for example, copying a disk file to tape as well as to disk pack) increases overhead. Most transfers occur in a single pass of the source file unless disk or disk pack files are transferred to mixed destination media (for example, copying a disk file to tape and to disk pack). In these cases, the source file is first transferred to the "like" medium and then to the "unlike" medium.

After all transferring is complete, the program closes each file according to its transfer type. That is, COPY and COPYADD commands release the source file, and MOVE commands remove the source file if no errors have occurred and a compare function is not pending. The system removes any duplicates from the destination files.

For tape files, if the RETAIN volume attribute is specified, the tape will not be dismounted at end-of-reel and end-of-volume. If the UNLOAD volume attribute is specified, the tape will be dismounted. If neither RETAIN nor UNLOAD is specified, by default the last reel of a source volume will be retained, and all reels of a destination volume will be unloaded.

The SYSTEM/COPY program performs the compare function if you request it to do so. Only those files that were transferred successfully are eligible for comparison. If you used a MOVE statement as the transfer function, the program removes the source file after it performs a successful comparison.

Finally, if you request the summary option, the SYSTEM/COPY program generates the ODT portion of the summary, listing only those files that were transferred successfully. Counts show the number of files transferred with and without errors. The SYSTEM/COPY program also generates a printed summary that duplicates the information that appears on the ODT. The printed summary includes more information on errors encountered and related operator responses.

Security

All security attributes that are associated with a given source file are copied to all destination files. Similarly, tape files carry their attributes with them when they are transferred to disk or disk pack.

If a file located on a source tape has corrupt security information, the SYSTEM/COPY program assigns PUBLIC/IO as its security use and type value when this file is transferred to any destination volumes.

Maximum Values

The following list shows the various maximum values that are associated with the current release of the SYSTEM/COPY program. The characters *mm* represent the mix number of the COPY program and the character *p* represents the system number.

- Syntax pass file (*mm*0*p*0—contains the input syntax): 40 segmentss/area, 20 areas = 800 lines.
- Directory pass file (*mm*0*p*0—contains directory information): 90 segments/area, 100 areas.

  The directory pass file contains file identifiers. The maximum number of file identifiers that can be contained in the directory pass file varies depending on whether or not you select files by date (CREATED, ACCESSED, UPDATED, and BACKEDUP options) and whether the source is a disk or pack file. The following table shows the limits.

If the source files are on. . .		then the maximum number of file IDs in the pass file is. . .
DISK	and you selected files by date	99999
DISK	and you did not select files by date	99999
PACK	and you selected files by date	17998
PACK	and you did not select files by date	99999

- Work file (W*mm*0*p*0—contains volume and file information): 1000 segments/area, 100 areas = 100,000 records.

  Record requirements vary with the number of volumes and files you specify and the number of file identifiers into which group specifiers are expanded. The maximum number of specified files for each volume is 999,999; the maximum number of expanded files for each group specifier is also 999,999.

- Tape directory file (T*mm*0*p*0—generated tape directory): 1500 segments/area, 100 areas = 150,000 records.
- File maximum depends on file name sizes.

- Print file (SYSTEM/COPY): Unlimited capacity if printer is used; backup capacity is governed by MCP limits.
- Destination volumes: You cannot specify more than 10 destination volumes. The number of source volumes is not limited.
- The maximum buffer size is 199800 digits. The buffer size multiplied by the number of buffers and then added to the program size of SYSTEM/COPY must not be greater than 1 million digits.

Date Handling

Various information about a file is stored in the file header. For diskpack files, this information includes the following dates:

- The date the file was first opened output or output/input (creation-date)
- The date the file was last opened input/output or extend (last-update-date)
- The date the file was last opened in any manner (last-access-date)
- The date the most recent backup copy of the file was made (last-backup-date). Specifically, this is the last time that SYSTEM/COPY, with BACKUPRUN SET, successfully transferred the file. (Before MCP 3.2.5, this was the date that a version 2 or later pack file was opened input by SYSTEM/COPY.)

These dates can be used to select files for copying, using the CREATED, ACCESSED, UPDATED, and BACKEDUP keywords. The file date selection criteria for SYSTEM/COPY are very versatile, but can be confusing. The following considerations will give some ideas on how to use date selection while avoiding mistakes.

Selecting the Right Date for Use with the Keyword

Care must be exercised in correlating the date and the meaning of the keyword. For example, the ACCESSED keyword refers to the file's last-accessed-date. Consider a file ABC that was accessed once on 3/25/94 and again on 3/31/94. This file's last-access-date is 3/31/94. But consider the following syntax:

```
COPY AND SET (ACCESSED BEFORE 03/30/94) = FROM TEST1(PACK) TO TESTA(TAPE)
```

The file ABC was indeed accessed before 3/30/94, namely on 3/25/94. But the file's last-access-date is 3/31/94, and so the file will not be copied.

Dates Are Inclusive

Dates used with the BEFORE, AFTER, and BETWEEN keywords are inclusive. For example, consider the following syntax:

```
COPY AND SET (UPDATED AFTER 02/28/94) = FROM TEST1(PACK) TO TESTA(TAPE)
```

This syntax selects files with a last-updated-date of 2/28/94 **and** later.

Similarly, consider the following syntax:

```
COPY AND SET (CREATED BETWEEN 94001 94002) = FROM TEST1(PACK) TO TESTA(TAPE)
```

This syntax selects files that were created on 94001 (1/1/94) **and** files that were created on 94002 (1/2/94).

When SYSTEM/COPY Examines File Dates

A masked file name (one that contains an equal sign) requires SYSTEM/COPY to examine the directories to determine which files to copy. In this case, SYSTEM/COPY obtains the date information for all the files at the same time.

When file names without masking characters are specified, SYSTEM/COPY examines the date information for each file as it is opened for processing.

Use of the TODAY Keyword

Potential windows can open when you use the TODAY keyword. Consider a daily request to dump files that were updated today:

```
COPY AND SET (UPDATED TODAY) TO DAILY(TAPE)
```

This syntax will miss files that update after SYSTEM/COPY completes its file selection process but before midnight. To cover this kind of condition, it may be best to provide a day of overlap in your request. For example, you could dump all files that have updated today or yesterday using a syntax like the following:

```
COPY AND SET (UPDATED AFTER TODAY - 1) TO DAILY(TAPE)
```

For Monday's daily backup, the weekend must be considered as well. To get the same kind of coverage as the previous example, the Monday night dump needs to regress three days to cover the period from Friday night through Monday night. For a Monday night dump, the syntax could be similar to the following:

```
COPY AND SET (UPDATED AFTER TODAY - 3) TO DAILY(TAPE)
```

How SYSTEM/COPY Affects a File's Dates

SYSTEM/COPY is the only program that can open a file without changing the file's dates (creation-date, last-access-date, and last-update-date as appropriate for the particular access). Also, the destination files made by SYSTEM/COPY inherit the dates of their source files. This means that the backed-up copy of a file has the same dates as the original copy of the file. Also, when the backed-up copy is reloaded to the system, it has the same dates as the original copy.

If BACKUPRUN is TRUE, a file's last-backup-date is updated. If BACKUPRUN is FALSE, the last-backup-date is **not** updated. You might want to set BACKUPRUN to TRUE while making archives, and to FALSE while making a casual copy of a file for nonarchival purposes. The default is FALSE.

The last-backup-date is set as of the start of execution of SYSTEM/COPY. If the execution spans midnight, files continue to be marked with the previous day's date, to provide consistency within the run.

The last-backup-date is changed only upon successful completion of file transfer. If an error occurs during a file transfer or compare, or if a file is skipped for any reason, the last-backup-date remains unchanged.

BNA File Transfer

For BNA facilities the SYSTEM/COPY program performs file transfers with the HOSTNAME attribute. There are several limitations necessary to implement the current BNA:

- The COPYADD function is not supported.
- Transfers involving tapes are not supported.
- The PCOPY function is not supported.
- The file BLOCKSIZE value must be less than 131,070.

The SYSTEM/COPY program will attempt to optimize the transfer by increasing the number of buffers for each file. The maximum number of buffers used is 9. When all HOSTNAMEs specified are recognized as the local host, the BNA network and associated transfer scheme will not be used.

Error Messages

When it encounters errors, the COPY program goes to an abnormal end-of-job. Error messages are always displayed on the ODT, regardless of the system or requested options. Consequently, an appropriate error message precedes any library maintenance messages that indicate an error has occurred. In general, the SYSTEM/COPY program attempts to pinpoint the syntax errors by displaying the text line in question and pointing to the word it was looking at when the error occurred. The following list contains brief explanations of the error messages the SYSTEM/COPY program can display.

```
BUFFERSIZE HAS 3600 DIGIT MINIMUM AND 199800 DIGIT MAXIMUM
```

For the BUFFERSIZE option, you specified a value that is not in the required range. BUFFERSIZE has a 3600 digit minimum and a 199800 digit maximum.

```
CAN'T COPY MIRRD. USERFL
```

A mirrored user file (USERFL) cannot be copied with SYSTEM/COPY. For the procedure for copying USERFL from one diskpack to another, refer to "Converting Version 1 Families to Version 2 or Greater Families" in "Pack Subsystems" in this manual.

```
COPY ABORTED; GROW REQ TOO BIG
```

A request for additional memory failed.

```
ADD FUNCTION ILLEGAL FOR BNA APPLICATION
```

The COPYADD function is not legal when a HOSTNAME attribute is present.

```
EXPECTED FILE LIST
```

The program expected a file ID or list of file IDs.

```
EXPECTED KIND ATTRIBUTE (DISK, PACK, TAPE, ICTAPE, OLDTAPE)
```

The program found a non-default volume identifier without a KIND attribute.

```
EXPECTED OPTION OR VALID OPTION ABBREVIATION (COMPARE, SUMMARY, ODTSUMMARY,
PRNSUMMARY, MPID, MANDATORY, SYNTAX, CHECK, BUFFERSIZE, BUFFERS, BACKUPRUN,
CREATED, ACCESSED, UPDATED, BACKEDUP, SPECIFY, ALLOCATED)
```

The program found an invalid option. SPECIFY appears in this message only if you used the keyword MERGE. ALLOCATED appears only if you are using MCPIX.

```
EXPECTED VERB (COPY, COPYADD, MOVE, COMPARE, DIR, MERGE)
```

The program found an invalid verb.

```
EXPECTED "TO"
```

The program expected the beginning of an output specification.

```
EXPECTED "TRUE" OR "FALSE"
```

A Boolean SET option argument was illegal.

```
EXPECTED "("
```

The program expected a left parenthesis.

```
EXPECTED ")" OR ","
```

The program expected a right parenthesis or comma.

```
EXPECTED ","
```

The program expected a comma.

```
INCOMPATIBLE FILE IDS FOR VOLUME KIND
```

You attempted to use on disk or disk pack media a multilevel file ID or an ID greater than 6 characters.

INVALID ATTRIBUTE

The program found an invalid or unsupported file or volume attribute.

INVALID ATTRIBUTE ARGUMENT

The value for this file or volume attribute is illegal.

INVALID COMBINATION OF ATTRIBUTES SPECIFIED

Not every attribute can be used with every other attribute or with every type of file. Check the input syntax.

INVALID DATE SPECIFIED

The valid date formats are mm/dd/yy, mm/dd/yyyy, yyddd, yyyyddd, TODAY, and TODAY – <number of days>.

INVALID FILE IDENTIFIER

A file identifier must be less than or equal to 6 characters in length and must begin with an alpha character or an acceptable special character.

INVALID NUMBER OF BUFFERS SPECIFIED (2<= N <= 10)

The range for the BUFFERS option is 2–10.

INVALID OPTION ARGUMENT

The specification for this option is illegal.

INVALID SYNTAX - EXPECTED ",", "EXCEPT", "FROM", OR "TO"

Another file or source specification, except specification, or destination specification was expected.

INVALID USE OF "ALLOCATED" OPTION

You have the ALLOCATED option, but you also specified a source as a destination. This specification is invalid.

INVALID USE OF "ORIGIN" AS A VOLUME KIND

You used origin either as a source kind or as a destination kind, and you did not supply any tape sources.

INVALID VOLUME IDENTIFIER

A volume identifier must have no more than 6 characters and must follow the same rules as a file identifier.

MAX BNA BLOCKSIZE ERROR

If you use the SYSTEM/COPY program with BNA, the file BLOCKSIZE value must be less than 131,070.

MISSING CLOSE PARENTHESIS

The program did not find matching parenthesis for a beginning parenthesis.

MISSING CLOSE QUOTE

The program did not find a closing quote for an opening quote.

MULTIPLE DATE SELECTION OPTIONS NOT ALLOWED

You can use only one of the date selection options at a time—CREATED, ACCESSED, UPDATED, or BACKEDUP.

NUMBER OF DESTINATION VOLUMES EXCEEDS MAXIMUM

You have exceeded the current limit of 10 output destinations.

"ON" EXPECTED

The program found invalid syntax for the DIR command. You must precede the specification of the volume with ON.

TAPE HARDWARE ILLEGAL FOR BNA APPLICATION

TAPE volumes are not supported across a BNA network.

<volume name><file name> GROW RQST TOO BIG

A request for additional memory has failed. Refer to "Recovery Options," in this section for more information about the choices that appear on your ODT.

Warning Messages

DATE OPTION IGNORED - NOT IMPLEMENTED

The DATE SET option is not implemented and its specification is ignored.

SAVEFACTOR ATTRIBUTE TOO BIG - 999 SUBSTITUTED

The maximum save factor permitted is 999 days.

UNSUPPORTED ATTRIBUTE IGNORED: attribute

The specified attribute is not supported.

Library Maintenance Messages

The SYSTEM/COPY program generates a library maintenance message for each file it attempts to copy. If you set the USE COPY option, these messages appear on the ODT.

If you use the SUMMARY option, a listing of all files copied appears on the ODT. Also, the program prints a list that contains the library maintenance messages. Figure 24–1 shows the format of these messages.

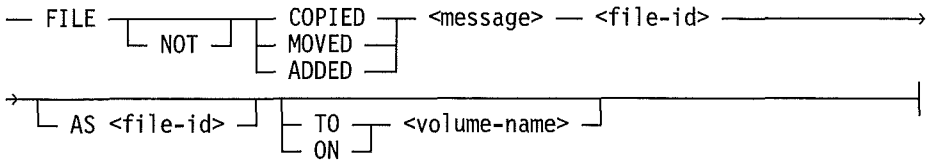


Figure 24–1. Library Maintenance Message Format

The TO and FROM volume names occur when you use the ORIGIN option for destination volumes.

The following paragraphs briefly describe each of the library maintenance error messages.

`DESTINATION ERROR`

An error has occurred on the destination volume. When this happens, the COPY program goes to an abnormal end-of-job.

`FILE ALREADY PRESENT`

This error occurs when you use COPYADD. A file has the same name as that of the file to be copied.

`FILE INCOMPLETE DUE TO PARITY ERROR`

During the operation for copying a file from tape, a parity error occurred. If you choose to use the CONTINUE command, the portion of the file that had been copied before the error occurred is left on the destination.

`FILE NOT COPIED-UPDATED BY MERGE: <file ID>`

The file identified was not copied because of a MERGE.

`FILE NOT COMPARED-UPDATED BY MERGE: <file ID>`

The file identified was not copied, and therefore not compared, because of a merge.

`FILE NOT PRESENT`

The requested file was missing from the source volume. If you set the MANDATORY option, the COPY program goes to an abnormal end-of-job.

INTERNAL ATTRIBUTE ERROR

An irrecoverable error occurred in the validation of the file header at file open time. The COPY program goes to an abnormal end-of-job.

NOT ON REQUESTED VOLUME INDEX

The file was not on the tape reel you requested. Because the tape directory contains all files residing on this and any subsequent reels of a volume set, the COPY program uses this message to indicate any files that did not begin on the reel you requested. The COPY program goes to an abnormal end-of-job.

OK

The stated function occurred without error.

VOLUME WAS ABANDONED

The relevant source volume was abandoned. Reasons for doing so vary (refer to "Error Messages" earlier in this section for more information). The COPY program goes to an abnormal end-of-job.

UPDATED BY MERGE

The program did not copy the requested file because the keyword MERGE was used. Another copy of this file was found on a subsequent source, so this file was updated by that file, and only that file was copied to the destination.

I/O Error Handling and Messages

When unrecovered or internal errors occur, the SYSTEM/COPY program displays a message on the ODT. An internal error is one that is recognized only by the SYSTEM/COPY program. A parity error is handled by the operating system first.

I/O errors are retried in accordance with established operating system guidelines. If recovery is not possible, control passes to the SYSTEM/COPY program for interpretation and possible action on your part.

Figure 24–2 shows the format of these error messages.

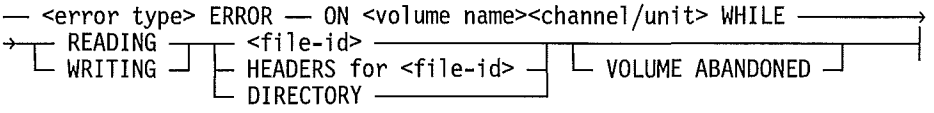


Figure 24–2. I/O Error Message Format

The <error type> construct defines the error. The <volume-name> construct specifies the volume on which the error occurred; when appropriate, the program also gives the <channel/unit> identifier.

The following paragraphs briefly explain each type of error.

COUNT ERROR

Each block on a SYSTEM/COPY tape is numbered to ensure that data items are read back in the proper sequence. This error message indicates a violation of that sequence. The error can be caused by an undetected read or write function, by dropped or added tape blocks, or by data corruption. When this error occurs, the COPY program goes to an abnormal end-of-job.

PARITY ERROR

An unrecoverable parity error occurred. The COPY program goes to an abnormal end-of-job.

PREMATURE EOF

The program encountered a premature end-of-file condition was encountered when it was reading or writing this file.

RELIABILITY ERROR

This error occurs when you are copying from a tape source that had an error when it was created. That error caused the file to be abandoned. The COPY program goes to an abnormal end-of-job.

Additional Error Possibilities

Both TAPE and ICTAPE volumes present additional error possibilities related to their headers, which precede the file and store information about the file. If an error occurs while the program is reading or writing this header information, the HEADERS FOR phrase notifies you.

Similarly, each tape volume has a directory near the front of each reel. Errors encountered while the program is reading or writing this information are signaled by the DIRECTORY message. If an error occurs while the Directory is being read, you have no recovery options. The volume is abandoned.

Recovery Options

In addition to an error message, the program may give you recovery options. The SYSTEM/COPY program judges the severity and recoverability of the error to determine the possibilities to offer. These options appear in this format after the error message and are in this format:

- AX "CONTINUE" to ignore error
- AX "QUIT" to abandon this file only
- AX "ABORT" to abandon this volume

You can abbreviate the response to the first letter, and the program checks it for validity.

The CONTINUE option instructs the SYSTEM/COPY program to ignore an error and to continue with the operation. If tape destinations are involved, the block containing the error is marked as unreliable. For this reason, a reliability error is noted if you use the volume block as input. Any disk or disk pack destination files are properly closed, although they are incomplete. You must determine the reliability of the file.

The QUIT option stops the transfer of this file and continues to the next file if one exists. Any destination tape files are marked, and a subsequent usage will result in a nontransfer for this file. Any disk or disk pack destination files are purged.

The ABORT option performs all the functions of the QUIT option and skips any remaining files requested from the current source volume.

Reliability Handling and Messages

Each block on a SYSTEM/COPY generated TAPE and ICTAPE contains reliability information. This information consists of a block count and a data integrity flag.

The block count begins at 1 and is incremented by 1 for each block of the file as it is written to the tape. When the program reads the tape, either during the compare phase or during subsequent use, it verifies this sequence will be verified. If the program finds a discrepancy, a COUNT error occurs.

The data integrity flag indicates the validity of the information contained in this block. If an error occurred on the source file when this tape file was created, you will be given various options. Your response is reflected in the value of the integrity flag written to the tape. When the tape block is read, the program interprets the value and produces the following message:

BLOCK NUMBER <number> ON TAPE <tape-name> CONTAINS UNRELIABLE DATA FOR SECTORS range OF FILE <file name>

This message tells you that an error occurred when the tape file was created. It gives the block number and sector range where the error occurred. A second message could appear, indicating the final outcome of this error.

Figure 24–3 shows the format of these messages.

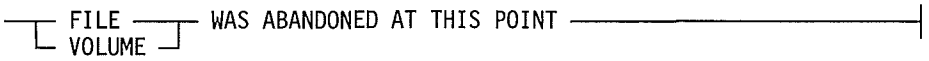


Figure 24–3. Block Error Message

If you respond to this message with a CONTINUE statement, the ABANDONED message does not appear.

The FILE WAS ABANDONED message indicates that you abandoned the file by using the QUIT response.

The VOLUME WAS ABANDONED message indicates that you abandoned this file and all files requested from that source volume by using the ABORT response.

The ABANDONED statement causes all destination files to be purged. Any tape destination files are marked with the same reliability information.

Caution

When you respond with a CONTINUE statement to a reliability error, take great care not to use the file before you investigate the cause of failure and repair the file as necessary.

Missing File Handling and Messages

The SYSTEM/COPY program might encounter a variety of missing file conditions during execution. Generally, these conditions are missing disk packs, missing disk or disk pack files, missing tape files, or duplicate disk and disk pack files. The following paragraphs describe each of them.

Missing Disk Packs

When you create destination tapes, you need to get directory information before any file transfer can occur. The SYSTEM/COPY program assumes the disk directory to be accessible, but a disk pack may be offline when its directory is needed. If this condition occurs, the SYSTEM/COPY program responds with the following messages:

```
<volume name> disk pack NOT ON LINE
```

You can mount the required disk pack or ignore the request for this source volume. The program displays two options as follows:

```
AX "CONTINUE" ONCE ON LINE, AX "ABANDON" TO ABANDON THIS VOLUME
```

You can abbreviate your response to the first letter, and the program checks it for validity. If the summary option is set, your response is reflected on the summary.

Missing Disk and Disk Pack Files

Missing disk and disk pack file messages can be sent when the program searches the directory or afterwards as the files are about to be opened for transfer.

If a file is missing or has been removed from the disk or disk pack after being located, the SYSTEM/COPY program sends the following message:

```
<file name> NOT FOUND ON <volume type> <volume name>
```

If the MANDATORY option was set, the SYSTEM/COPY program displays the following message:

```
<count> FILE(S) NOT FOUND ON <volume type> <volume name> PLEASE AX TO
ACKNOWLEDGE AND CONTINUE
```

The SYSTEM/COPY program halts until you acknowledge that some files have not been found. If you respond with an AX command, the SYSTEM/COPY program continues with the next file.

Missing Tape Files

When working with source tape volumes, you must enter the directory and search it for all requested files. If you request more than one file for a destination of disk or disk pack only, the order of the files might be altered to coincide with their order on the directory to avoid rewinding the tape. As with disk and disk pack files, the SYSTEM/COPY program ensures that requested files are present and notifies you of their absence.

When a tape is created, a file can appear in the tape volume directory without actually being downtape. An example of this would be a parity error that was on the tape when it was created, which was located after the directory but before the file was transferred. If the volume was abandoned at that point, the file would be in the directory of the tape but not downtape. The SYSTEM/COPY program notifies you of this situation with the following message:

```
FILE NOT FOUND DOWNTAPE ON TAPE <tape name>: <file name>
```

Duplicate Disk and Disk Pack Files

If you copy to disk or disk pack when a file with a matching ID is already present, the existing file is replaced with the new file. If the existing file is in use when the new file is copied, the existing file is removed when it is no longer in use. If another person tries to access the existing file while it is still in use, the person receives the newly copied file, not the existing, or original file already on disk or disk pack.

Nonlibrary Tape Handling and Messages

If you provide tape as input to the SYSTEM/COPY program and if the tape is not a SYSTEM/COPY, LOADMP, or PACKUP format tape (that is, if it is a tape created by DMPALL or is a "foreign" tape), then the following message appears and the SYSTEM/COPY program goes to an abnormal end-of-job.

```
<volume name> (cc/uu) IS NOT SYSTEM/COPY, LOADMP OR PACKUP TAPE-ANALYSIS
ABANDONED
```

LOADMP/PACKUP Tape Handling and Messages

If you use the VOLUMEINDEX option with a LOADMP or PACKUP tape, then the following message appears and the SYSTEM/COPY program goes to an abnormal end-of-job.

```
<volume name> (cc/uu) IS A LOADMP OR PACKUP TAPE VOLUMEINDEX SPECIFICATION
IS NOT ALLOWED-ANALYSIS ABANDONED
```

ICTAPE Format

The ICTAPE format follows the Unisys Corporate Standard format for interchangeable media. At this time however, neither B 1000 Series nor the A Series implement this standard. It is a multifile tape format. The general format of an ICTAPE is the following:

```
VOL1...file(s)
```

VOL1 represents the volume label written to the tape by the MCP. In this case, a file is of this general form:

```
HDR1...HDR2...TM...FID...data block(s)...TM...EOF1...EOF2
```

HDR1 and HDR2 are 80-byte records that are written by the MCP when the file is opened. The MCP receives the information for the HDR1 and HDR2 records from the SYSTEM/COPY program. These records do not contain the file header information that is kept in a disk or disk pack file header. The HDR2 record contains the maximum blocksize that occurs on the tape.

TM represents a tape mark.

FID, or file description block, contains the information found in a disk or disk pack file header. Both an object header (system dependent) and a symbolic header (system independent) are contained in the FID.

Unless you use the BUFFERSIZE option to override this default, the FID and all data blocks are 9KB blocks plus 12 bytes of block information. The last block of a file cannot have a full 9KB of valid data, but the program still writes a 9KB block. Actual file end-of-file (EOF) must be determined from the file header information in the FID.

EOF1 and EOF2 are 80-byte records that are written by the MCP when the file is closed, and they are essentially duplicates of the HDR1 and HDR2 records.

Directory Format

The first complete file on a tape is the DIRECTORY. The following is the format of the DIRECTORY:

```
HDR1...HDR2...directory block(s)...EOF1...EOF2
```

where the first directory block consists of

```
FORMAT ENTRY...FILE ENTRIES
```

Any subsequent directory block or blocks consist of only file entries. A file entry of all zeros is the last entry.

Each block that is written to the tape contains a 12 byte Block Information Field. This field is referred to as the Record Information field or RIF. Because of this field, each block is actually 9012 bytes; the first 12 bytes are this field. These 12 bytes are identical to the first 12 bytes of the RIF for a TAPE format tape.

Table 24–1 shows the information in these 12 bytes.

Table 24–1. ICTAPE Format RIF

Pos (UN)	Field Name	Len/Type	Content
1–20	Transaction number	10UA	This field is a count on all data blocks in a file, starting with one for each file.
21–22	Block type	1UA	This field is a flag indicating the type of block: 1 = directory 2 = FID 3 = data
23–4	Reliability	1UA	This field indicates the reliability of the data contained in this block.

Reading An ICTAPE

A program must specifically allocate an extra label area large enough to accommodate the USASI Standard label that an ICTAPE has, that is, 28 bytes for a basic label area and 80 bytes each for the HDR1 and HDR2 records.

Opening the tape file with the multifile ID set to the tape name and the file ID set to blanks causes in the first file on the tape to be opened. This file ID is TAPDIR for reel one, or the continuation of the file from the previous reel for any subsequent reels of a multireel set.

For any reel beyond reel 1, the first file ID on the tape is Fxxxxx, where xxxxx is a file sequence number relative to the first file on reel 1. This file is continued from the previous reel onto this reel. The next file is TAPDIR.

Each file on the tape is a separate file. Therefore, subsequent readings will read through the file until EOF is sensed. Once the EOF is sensed, the file can be closed. If the file ID is reset to blanks, the next open will yield the next file. If you want a particular file, you can then request that file ID, and the MCP searches the tape until it finds the requested file.

You should perform open and close files with the no-rewind option if you want to read through the entire tape. A simple closing rewinds the tape, but leaves the program attached. A close release rewinds the tape and returns the unit to the system.

TAPE Format

The TAPE format is somewhat similar to the LOADMP/PACKUP tape format in that it is a single file tape. The general format of a TAPE is the following:

```
VOL1...HDR1...HDR2...TM...file(s)...TM...EOF1...EOF2
```

where a file is of the general form

```
FID...data blocks
```

VOL1 represents the volume label written to the tape by the MCP.

HDR1 and HDR2 are 80-byte records that the MCP writes when the file is opened. The MCP gets the information for the HDR1 and HDR2 records from the SYSTEM/COPY program. These records do not contain the file header information that is kept in a disk or disk pack file header. The HDR2 record contains the maximum blocksize that occurs on the tape.

TM represents a tape mark.

The file description block (FID) contains the information that is found in a disk or disk pack file header. Both an object header (system dependent) and a symbolic header (system independent) are contained in the FID.

The FID and all data blocks that are not a full block are written as short blocks to the tape. Though the exact size of a given data block is not stored, data block size can be determined from FIBRAD after the block is read. Actual file EOF must be determined from the file header information in the FID.

EOF1 and EOF2 are 80-byte records that are written by the MCP when the file is closed. These records are essentially duplicates of the HDR1 and HDR2 records. The first file is a directory and has the general format:

```
directory blocks
```

where the first directory block consists of:

```
FORMAT ENTRY...FILE ENTRIES
```

Any subsequent directory blocks consist only of file entries. A file entry of all zeros is the last entry.

Directory blocks are also written as short blocks if they are not a full 9KB.

The definitions of these fields are identical to the description in this document, with the following exception: the fifth byte in the HDR2 record is a U, which designates variable length records, where length is determined by each I/O statement.

Each block written to the tape contains a 24-byte Record Information field. This is referred to as the RIF. Because of this field, each full block is actually 9024 bytes, where the first 24 bytes are this field.

Table 24–2 shows the information in these 24 bytes.

Table 24–2. TAPE Format RIF

Pos (UN)	Field Name	Len/Type	Content
1–20	Transaction number	10UA	Counts all data blocks in a file, starting with 1 for each file
21–22	Block type	1UA	Flags the type of block 1 = directory 2 = FID 3 = data
23–24	Reliability	1UA	Indicates the reliability of the data contained in this block
25–36	Checksum field	12UN	Used explicitly by DMCOPY (DMSII)
37	Checksum flag	1UN	Used explicitly by DMCOPY (DMSII)
38	DMCOPY tape type	1UN	Used explicitly by DMCOPY (DMSII)
39–48	Filler	10UN	Reserved for future use

Reading a TAPE

A TAPE is one continuous physical file. Different block types have been defined to logically differentiate one file from another. These block types are designated in a field in the RIF (see block type in Table 24–2).

A program must specifically allocate extra label area large enough to accommodate the USASI Standard label that a TAPE has, that is, 28 bytes for a Basic Label Area and 80 bytes each for the HDR1 and HDR2 records.

Opening the tape file with the multifile ID set to the tape name and the file ID set to blanks causes the only file on the tape to be opened. This file ID is TAPDIR for reel one, or Fxxxxx, where xxxxx is a file sequence number, for any reel beyond reel 1. Fxxxxx indicates this reel is not reel 1 and that the first logical file is a continuation of the file from the previous reel. The next logical file is TAPDIR.

After the file is opened, the first reading yields a type 1, or directory block (for reel 1). Because there can be more than one directory block, the end of the directory can be determined by reading type 2 block (FID).

The FID is followed by any number of type 3 data blocks. When another type 2 block is read, then the program can determine the end of the previous file. This procedure is repeated for each file on the tape. The end-of-file for the last file is determined by an actual EOF on the read statement.

Determining the Format of a TAPE Programmatically

You can differentiate ICTAPE, TAPE, LOADMP, and PACKUP tape formats by first opening the tape.

The program must allocate extra label area large enough to accommodate the USASI standard label that a TAPE and an ICTAPE have, that is, 28 bytes for a basic label area, and 80 bytes each for the HDR1 and HDR2 records. LOADMP and PACKUP tapes have a standard label.

The tape file must be opened with the multifile ID set to the tape name and the file ID set to blanks. This causes the first file on the tape to be opened.

When the file opened, the label area is filled.

If the HDR1 and HDR2 records are not filled in, then the tape is a LOADMP or PACKUP tape. The eighth byte of the basic label area differentiates the tape format. For LOADMP it is 5, and for PACKUP it is 6.

If the HDR1 and HDR2 records are filled in, then the tape is a TAPE or an ICTAPE. The fifth byte of the HDR2 record differentiates the tape format. For a TAPE, this byte is a U, and for an ICTAPE it is an F.

Determining the Format of a Tape Visually

For a TAPE, the only physical file on the tape is named TAPDIR. For an ICTAPE, the first file is TAPDIR. For any reel other than reel 1 of a multireel set of tapes, the first file on an ICTAPE has a file ID of Fxxxxx, where xxxxx is a file sequence number relative to the first file on reel 1. This file is continued from the previous reel onto this reel.

The only file on a LOADMP or PACKUP tape is named FILE.

If you enter OL MTP on the ODT when a tape is mounted and ready, the multifile ID and the first file ID appears. For TAPE and ICTAPE format tape, the response is the following:

```
<multifile ID>/TAPDIR for SYSTEM/COPY TAPE or ICTAPE

<multifile ID>/Fxxxxx for SYSTEM/COPY ICTAPES (reel > 1)
```

For LOADMP/PACKUP tapes, the response looks like this:

```
<multifile ID>/FILE for LOADMP/PACKUP tapes
```

To determine the exact format of a tape, TAPE, ICTAPE, LOADMP, or PACKUP, you should use the SYSTEM/COPY DIRECTORY command to get a directory listing of the tape. The printer backup file generated will describe the exact tape format.

Section 25
SYSUP—Automatic System Recovery Facility

Overview

The Automatic System Recovery facility (SYSUP) provides a means to automatically recreate the operating environment after a halt/load or system failure. The facility is specifically advantageous for systems that fail in an unattended, online environment. When the system fails, the SYSUP facility can provide information regarding the type of halt/load and the task number of the job being executed at the time. Also, it can execute a user program that performs such tasks as executing programs, readying printers, and recreating the operating environment.

The MCP can initiate the SYSUP facility automatically after any type of halt/load. If a user version of SYSUP is available, the MCP executes it; otherwise, the MCP executes a version bound to the MCP.

The MCP passes two digits of information to the SYSUP program when it is initiated. The first digit describes the type of halt/load that took place, the second digit contains the system number.

A user SYSUP program can use the data that the MCP passes to SYSUP. A user version tends to be more sophisticated than the version bound to the MCP to suit the needs of your installation. The bound version of the SYSUP facility is fairly simple. It zips a command to the MCP and goes to end-of-job. On the other hand, user versions of the SYSUP facility can do such things as make disk packs ready, set up printers, dump MCP memory, and bring up timesharing and handlers.

When you enter the maintenance processor command LOAD MCP, a halt/load occurs.

- If you include a USE record with the AUHL option in the system configuration file, an automatic halt/load occurs after the system fails.
- If you include a USE record with the SYSUP option, the SYSUP facility is executed automatically. For more information about these USE records and options, refer to Volume 1.
- If you set the AUHL option and a halt/load occurs, the bound SYSUP program sends a command to the MCP to start the DMPANL program to analyze the MCP memory dump file.

The SYSUP facility stores two digits of information in memory at base-relative addresses 32–33. The following paragraphs explain the meaning of each digit.

The first digit indicates the type of halt/load. It can have the following values and meanings:

Digit	Type of halt/load
0	Manual halt/load
1	ODT halt/load
4	Cold-start halt/load
9	Automatic halt/load

The second digit indicates the system number.

SYSUP Programming Considerations

You usually need a full dump to determine why the system failed. Therefore, you may want your version of the SYSUP facility to execute the DMPANL program with a ZIP PM 1 request. You should then wait for the DMPANL program to finish before you start any jobs or handlers. You might want to print out the MCP table (TBL) dump for immediate use or copy the dump file ($p0001) to tape or disk pack for later analysis.

Example SYSUP Programs

These are examples of SYSUP programs. Example 25–1 illustrates the bound SYSUP program. Example 25–2 illustrates a user SYSUP program that will restart an unattended timesharing system.

```
BEGIN
   CONTROL STACK:= 0& FLAT PROGRAM
   INTEGER WHOCALLS (1) = 32,
           PROCNUM  (1) = 33;
   ALPHA DUMPER (16) :=  "SPO PM1"
   IF WHOCALLS = 9 THEN
   BEGIN
       ZIP DUMPER;
   END;
END;
```

Example 25–1. Bound SYSUP Program

```
SYSUP:
   BEGIN
      CONTROL STACK := 0;& FLAT PROGRAM;

      INTEGER WHOCALLS (1) = 32,
              PROCNUM (1) = 33,
              PROGRAM_RUNNING (2);& TO SEE IF JOB IS RUNNING;

      ALPHA DUMPER (8) := "SPO PM1." ;

      & FIRST READY ALL OF THE DISK PACKS & IN CASE H/L SAVED THEM;
      ZIP "SPO RY 8/0. " ;
      ZIP "SPO RY 8/1. " ;
      ZIP "SPO RY 8/2. " ;
      ZIP "SPO RY 8/3. " ;

      & NOW SET UP THE PRINTERS CORRECTLY;
      ZIP "SPO RY7/0 DOC96. " ; & WHITE PAPER;
      ZIP "SPO RY 17/0 SHORT. " ; & SHORT BLUE PAPER;
      ZIP "SPO RY 15/0 LINED. " : & GREEN STOCK FORMS;
      ZIP "SPO RY 19/0 SPEC. " ;& DOCUMENTATION PRINTER;

      & SAVE THE SPECIAL FORMS PRINTERS;
      ZIP "SPO SV 17/0. " ;
      ZIP "SPO SV 15/0. " ;
      ZIP "SPO SV 19/0. " ;

      & IF AUTOMATIC HALTLOAD DO A DUMP FIRST;
   IF WHOCALLS = 9 THEN& AUTOMATIC H/L?
   BEGIN
      ZIP DUMPER;
      & WAIT FOR DMPANL TO FINISH;
      DO BEGIN
         DOZE 10;
         PROGRAM_RUNNING := TASKID "DMPANL " ;
      END UNTIL PROGRAM_RUNNING = 0;
   END;& END OF SYSTEM DUMP CODE;

      & BRING UP THE SYSTEM MONITOR (IF NOT ALREADY UP)

      PROGRAM_RUNNING := TASKID "SYSMON " ;
      IF PROGRAM_RUNNING EQL 0 THEN
         ZIP "EX SYSMON PR 6 MEM 13 LOCK. " ;

      & AND NOW TO BRING UP TIME-SHARING;

      ZIP "EX CTLMCS PR 7 LOCK. " ;
   END;
```

Example 25–2. User SYSUP Program

Section 26
UNLODV—Uniline DLP Utility Program

Overview

Use the UNLODV program for Uniline DLPs that connect the system to different kinds of data terminal equipment (DTE) and data communications equipment. This equipment includes ODTs, Remote Job Entry terminals, and teletypewriters. Other devices include various kinds of RS232 equipment and modems.

The UNLODV program matches the data communications parameters that the Uniline DLP uses to those that the device attached to that Uniline DLP uses. The program sets these data communications parameters in the form of firmware, and can either save the firmware in a file or load it to the DLP. If saved in a file, the firmware can be loaded to the Uniline DLP later with the LH command or with the LOADFW or UNLODV programs.

The data communications parameters must be matched to establish the first level interface, which is the electrical connection used to exchange and check signals between the Uniline DLP and the device. If these parameters do not match, the Uniline DLP and the device will probably not communicate.

Two different files store the basic firmware that UNLODV reads in and modifies. The files are named USP3BH and UST3BH. The file that you use depends on the classification of equipment that is attached to the Uniline DLP. These basic firmware files are distributed on the system software release tape and on the UNILINE support files tape, which are included in the UNILINE test and field package.

The UNLODV program customizes the firmware in memory in accordance to the commands that you enter. Then the UNLODV program either loads the firmware into the Uniline DLP, stores the customized firmware in a file that you name, or both. You can load the customized firmware file into the Uniline DLP later.

The following paragraphs describe the two basic firmware files.

Firmware File USP3BH

This file contains ODT look-alike firmware. Use this file if the device connected to the Uniline supports a secondary ODT connected with an RS232 asynchronous direct connection. This file supports the following equipment: ET, MT 983, TD8, T 27, TD 830, TC 4000.

The USP3BH settings are the following:

ASCII	Setting
BAUD RATE	09600
ADAPTER-IO	OO
ASYNCHRONOUS	SET
EIA	SET
83900	SET
UNILINE	2AC
CHAR-LENGTH	7 BITS
DIRECT CONNECT	SET
TIMEOUT-EOT	RESET
RETURN CR AS:	CR
RECEIVE-DELAY	00001
TRANSMIT-DELAY	00100
SQUELCH-DELAY	00000
TIME-CUT	09999
STOPBITS	1

Firmware File UST3BH

This file contains standard terminal control look-alike firmware that you use for various types of data terminal equipment (DTE) and modems. You can use it to connect the Uniline to a network of terminals using poll/select or to communicate system-to-system either with a direct connect or a telephone system. This file runs RS232, synchronous or asynchronous, or TDI with or without modems. The connection can be with switched (dial) or leased lines.

The UST3BH settings are the following:

ASCII	Setting
BAUD RATE	09600
ADAPTER-IO	00
ASYNCHRONOUS	SET
EIA	SET
83900	SET
UNILINE	2AC
CHAR-LENGTH	7 BITS
DIRECT CONNECT	SET
TIMEOUT-EOT	RESET
RETURN CR AS:	CR
RECEIVE-DELAY	00001
TRANSMIT-DELAY	00002
SQUELCH-DELAY	00016
TIME-CUT	05000
STOPBITS	1

You can display the setting of the data communications options in a file:

1. Execute the UNLODV program and load the file in memory.
2. Enter the DISPLAYPARA command to display the settings on the screen.

System Configuration Records

You can use two different types of system configuration records to declare Uniline DLPs. You must use a DLP record to declare what channel the Uniline DLP is on. Then you must use a UNIT record to declare the type of device that is connected to the Uniline DLP. The following system configuration records declare Uniline DLPs and devices to the MCP:

- DLP record
- UNIT RJE record
- UNIT TC5 record
- UNIT VDD record
- UNIT TWX record

Refer to Volume 1 for more information about these records.

Executing UNLODV

You can execute UNLODV from an ODT or from a card reader. If you execute the program from an ODT, enter the commands and data communications parameters with the AX command. You can enter only one command with each AX command.

If you execute the program from a card reader, enter the commands and the data communications parameters when you execute it.

When you have executed the UNLODV program, you can customize firmware for one Uniline DLP while it loads firmware to another Uniline DLP.

Use the following syntax to execute the UNLODV program from an ODT:

```
EX UNLODV (,"<name of file with the basic firmware>", "S")
```

If you execute the UNLODV program from a card reader, you can enter only one command or parameter per record. The program expects a data card with the name COMFIL.

Use the following syntax to execute the UNLODV program from a card reader:

```
?EX UNLODV (,"<name of file with the basic firmware>")
?DATA COMFIL
TD8
MAKE FILEAA
?END
```

Many of the data communications parameters have value ranges. You must enter a value for each parameter that has a value range.

When you execute the UNLODV program, write down the mix number. You need the mix number to enter data communications parameters and commands to the program. The following are examples of the syntax you use to enter AX commands and parameters:

```
< mix number> AX  "TD8"
< mix number> AX  "MAKE FILEAA "
```

Data Communications Options

The following table lists the data communications options and their values, if appropriate. If a value appears, you must enter a value. The default value for each file appears under the discussion of the TD8 option in this section.

Data Communications Options	Value Range
ADAPTERID	<hex number 0–1F>
ASCII	
ASYNCHRONOUS	
BAUD	<110–19200>
BDI	
CHARACTERLENGTH	<5–8>
CONVERTCR	
DATASET	
DIRECTCONNECT	
EBCDIC	
EIA	
LEASEDLINE	
PARITY	<ODD, EVEN, or DISABLED>
POLLTIMEOUTEOT	<SET or RESET>
PUSH	<0–4>
RECEIVEDELAY	<0–255>
SQUELCHDELAY	<0–255>
STOPBITS	<1, 1.5, or 2>
SWITCHEDLINE	
SYNCHRONOUS	<SINGLE or DOUBLE or none>
TD8	
TDI	

continued

Data Communications Options	Value Range
TIMEOUT	<number of milliseconds 1–9999>
TRANSMITDELAY	<number of milliseconds 0–255>
U2AC	
U2B	

RECEIVEDELAY and SQUELCHDELAY are additive on any I/O (including poll and transparent options) that contains WRITE-FLIP READ.

You can shorten any option so long as the abbreviation is unique from other options. For example, you can abbreviate ASYNCHRONOUS to ASY but not to AS, because AS could also designate ASCII.

The following pages describe each of the data communications options.

ADAPTERID <hex number 0–1F>

This option sets the adapter ID in a standard terminal control. The program uses the adapter ID only in the result descriptor of a TEST I/O operation, which is generally used by remote job applications (RJE).

ASCII

This option makes the Uniline sensitive to ASCII control codes. The option must match the ASCII/EBCDIC hardware strap.

ASYNCHRONOUS

This option makes the transmission and reception of data asynchronous.

BAUD <110–19200>

This option sets the BAUD rate for the transmission and reception of data. Use this option only when ASYNCHRONOUS is set.

The BAUD rates for the Uniline type 2B range from 110 to 19200.

The BAUD rates for the Uniline types 2, 2A, and 2C range from 110 to 9600.

BDI

This option enables the BDI interface.

CHARACTERLENGTH <5–8>

This option sets the number of data bits per transmitted and received character.

CONVERTCR

This option causes a received CR code (0D hex) to be converted to an ETX code (03 hex) during a READ CONTROL operation, the read portion of a WRITE FLIP TO READ CONTROL operation, or the positive response to a POLL operation.

DATASET

This option describes the remote device that is connected to the Uniline with a data set to a switched (dialed) line.

DIRECTCONNECT

This option describes the remote device that is connected directly to the Uniline DLP (for example, without an intervening data set). You can specify TDI or BDI separately.

EBCDIC

This option makes the Uniline sensitive to EBCDIC control codes. This option must match the ASCII/EBCDIC hardware strap.

EIA

This option is the same as RS232.

LEASEDLINE

This option indicates that the Uniline is attached to a leased line data set.

PARITY <ODD, EVEN, or DISABLED>

This option describes the parity generation and checks for each character transmitted and received on the data communications line.

POLLTIMEOUTEOT

This option causes a timeout on a RECIRCULATING POLL I/O OPERATION to be treated as a negative response instead of returning a timeout result descriptor.

PUSH <0–4>

This option represents the number of push characters (FF hex) appended to each synchronous transmission. Use this option to "push" the last data characters through the sync data sets.

RECEIVEDELAY <0–255>

This option sets the delay in milliseconds between the detection of the carrier detect signal and the beginning of the firmware search for valid data.

SQUELCHDELAY <0–255>

This option sets the delay in milliseconds between the last bit being transmitted and the next event. This value applies only to a DATASET device; set it to zero for a direct connect line.

STOPBITS <1, 1.5, or 2>

This option sets the number of stopbits transmitted or expected for each character transmitted or received. Use this option only for asynchronous transmissions.

SWITCHEDLINE

This option has the same meaning as the DATASET command and indicates that the UNILINE is attached to a switched (dial) line data set.

SYNCHRONOUS <SINGLE, DOUBLE or none>

This option indicates that the transmission or reception of data is synchronous and the number of sync characters to expect before synchronization is achieved. Because sync characters are always expected on a synchronous line, you should use SYNCHRONOUS DOUBLE.

TD8

This option sets the parameters for the firmware (refer to the following tables). For Uniline 2, use command TD8 and Baud to set Baud rate.

ASCII	Asynchronous
BAUD	9600
CHARACTERLENGTH	7
DIRECT CONNECT	
PARITY	EVEN
RECEIVE-DELAY	2
TRANSMIT-DELAY	10
STOPBITS	1
TIME-OUT	5000

The following are the USP3BH settings with TD8.

ASCII	Setting
BAUD RATE	05907
ADAPTER-IO	OO
ASYNCHRONOUS	SET
EIA	SET
83900	SET
UNILINE	2AC
CHAR-LENGTH	7 BITS
DIRECT CONNECT	SET
TIMEOUT-EOT	RESET
RETURN CR AS:	CR
RECEIVE-DELAY	00002
TRANSMIT-DELAY	00010
SQUELCH-DELAY	00000
TIME-CUT	05000
STOPBITS	1

TDI

This option enables two-wire direct interface hardware.

TIMEOUT <1–9999>

This option sets the number of milliseconds to wait for receiving a character before timing out.

TRANSMITDELAY <number of milliseconds 0–255>

This option sets the number of milliseconds to delay between receiving the clear-to-send signal and transmitting the data.

U2AC

This option sets the firmware to that used by the Uniline types 2A or 2C.

The following are the USP3BH settings with U2AC.

ASCII	Setting
BAUD RATE	09600
ADAPTER-IO	OO
ASYNCHRONOUS	SET
EIA	SET
83900	SET
UNILINE	2AC
CHAR-LENGTH	7 BITS
DIRECT CONNECT	SET
TIMEOUT-EOT	RESET
RETURN CR AS:	CR
RECEIVE-DELAY	00001
TRANSMIT-DELAY	00100
SQUELCH-DELAY	00000
TIME-CUT	09999
STOPBITS	1

U2B

This option sets the firmware to that used by the Uniline type 2B.

The following are the USP3BH settings with U2B.

ASCII	Setting
BAUD RATE	09600
ADAPTER-IO	OO
ASYNCHRONOUS	SET
EIA	SET
83900	SET
UNILINE	2B
CHAR-LENGTH	7 BITS
DIRECT CONNECT	SET

continued

ASCII	Setting
TIMEOUT-EOT	RESET
RETURN CR AS:	CR
RECEIVE-DELAY	00001
TRANSMIT-DELAY	00100
SQUELCH-DELAY	00000
TIME-CUT	09999
STOPBITS	1

UNLODV Commands

When you execute the UNLODV program and enter the data communications parameters to modify the firmware, use the following commands to save the customized firmware in a file, to load the firmware to a Uniline DLP, and to patch the firmware object code.

END

This command ends the UNLODV session.

DISPLAYPARA

This command displays the data communications settings of the firmware file.

LOAD <Uniline DLP channel number>

This command loads the modified firmware from memory to the Uniline DLP that is connected to the specified channel.

MAKE <file-name>

This command tells UNLODV to store the firmware that has been customized in memory in a file with the given name. Load this file to the Uniline DLP later with the LH, UNLODV, or LOADFW command.

PATCH

This command permits modifications to the actual object code in the firmware. The syntax for PATCH is the following:

PATCH V<version number> <hex address> <hex data>

The data can be up to 60 characters in length. For example, if the firmware has the hex data 0123456789 patched into it at hex address 0FA8, the input command is as follows:

```
PATCH V601 0FA8 0123456789
```

Firmware File Format

The UNLODV program stores firmware in a single-area firmware file blocked 100 bytes by 1. The first record contains information on the size of each load phase. There can be up to three load phases although only the first phase is applicable to the Uniline. following table represents the first record in the file. It specifies the size of each phase. Zero indicates that the program has skipped that phase.

Offset	Size/Type	Description
0	6 UN	Phase One Load Size
6	6 UN	Phase Two Load Size (Zero for this program)
12	6 UN	Phase Three Load Size (Zero for this program)
18	182 UN	Zero Fill

All other records in the file contain 100 bytes of load data each. The file resides on disk.

Loading Firmware to Uniline

To load firmware to a Uniline from a V 300, refer to the *V 300 Maintenance User's Guide.*

To load firmware to a Uniline from a V 500, refer to the *V 500 Maintenance Reference Manual.*

Section 27
DCP—Data Communications Processor

Overview

A Data Communications Processor (DCP) enhances the data communications capabilities of the V Series systems. A DCP reduces the workload on the host system by performing data communication tasks that the host would otherwise have to handle. Some of these tasks include routing messages to terminals, recovering from errors, translating code, managing lines, and managing networks.

System Requirements

A data communications system that uses B 874, B 974, TELCOM, or ICPV (CP 2000) DCPs requires a message control system (MCS), a network definition language (NDL), and the DCP module of the MCP.

MCS

The MCS enables you to manage various aspects of the data communications system, such as security, file control, error handling, audit and data recovery, message routing to applications programs, and resource allocation. Unisys can supply an MCS (for example, GEMCOS), or you can write your own in a high level language such as COBOL or BPL. Refer to the *B 874/B 974/ORS-DLP MCS Message Headers Programming Reference Manual* for more information.

The required B 874 or B 974 Network Definition Language (NDL) source program includes a concise description of the network and of the procedures you should follow to control the network. Normally, you would write the NDL source so that it meets your local network requirements. This NDL source is compiled with the B 874 NDL compiler (CPNDL1) or the B 974 NDL compiler (NDL974). The compilation process creates the following files: one NDL firmware file for each DCP, an MCP network information file (MCPNIF), and an MCS network information file (MCSNIF). These files reside on disk on the host system.

DCP

The MCPNIF must be present on disk when you set the required DCP module on the host system. The NDL programmer selects the name of the firmware file. The names of the NIF files are MCPNpF and MCSNpF where the p represents the system number of the host

system that will use the files. If you want a B 874 and a B 974 DCP to run simultaneously on the same system, you must use the NIFMRG program to merge the NIF files.

You must load the firmware file into the DCP before it can operate. There are two commands you can use to load the firmware file into the B 874 or B 974: the LH command and the SO command with the DCP option. Refer to Volume 2 for more information about these commands.

Network Initialization

Use the following procedure to initialize a data communications network to run a B 874 or a B 974 Data Communications Processor.

1. Perform a cold-start on the system and be sure to declare the channel for each DCP. Each DCP must have its own UNIT card. Refer to Volume 1 for more information about the UNIT record with the DCP option.
2. Generate a firmware file for each DCP and the MCPNIF and MCSNIF files, using the appropriate NDL compiler. A B 874 and a B 974 DCP cannot run simultaneously on the same system unless you use the NIFMRG program to merge the NIF files.
3. Enter the SO command with the DCP option. An LH command occurs automatically when you use the DCP option and the SO command.
4. Execute the MCS when the LDHOST program goes to end-of-job or the B 974LD program displays the following message:

```
DATA COMM TO STATIONS HAS BEEN SUCCESSFULLY ACTIVATED
```

MCP Interfaces

The following MCP records and commands initialize and control B 874, B 974, TELCOM or ICPV DCPs. For more information about UNIT and LIMIT records, refer to Volume 1. For more information about commands and control instructions, refer to Volume 2.

DLP Record

The DLP record declares a DLP for the DCP.

DCP UNIT Record

The DCP UNIT record configures the system for a DCP.

LIMIT DCPQUE Record

The LIMIT DCPQUE record sets the maximum WRITE functions (type 30) that can be outstanding in the DCP for each station.

LIMIT DCPBUF Record

The LIMIT DCPBUF record sets the default number of input messages that an MCS result pool can store.

SO Command with the DCP Option

The SO command with the DCP option sets the DCP option and loads the DCP module into the host system memory. The command executes the LDHOST program for each B 874 and the B 974LD program for each B 974.

LH Command

The LH command loads firmware and initializes DCPs.

BUFFER Control Instruction

The BUFFER control instruction specifies the number of input messages an MCS result pool can store. BUFFER overrides the value specified for a specific MCS in the LIMIT DCPBUF Record.

Table 27–1 lists B 874 error conditions.

Table 27–1. B 874 Link Errors

INV R/D	R/D = <result-descriptor>
CONTROL TIMEOUT	R/D = F000
HTMALFUNCTION	R/D = F800
B 874 MALFUNCTION	R/D = F700
S-MEM PARITY	R/D = CC00
MTEB	R/D = C400
MTEX	R/D = C200
HTLINK PARITY	R/D = D100
B 874 LINK PARITY	R/D = D200
INV S-MEM ADRS	S-PTR = <S-mem-address>
INV FUNCTION TYPE	FUNC = <function-number>
INV PSN	PSN = <PSN-value>

Table 27–2 lists B 974 error conditions.

Table 27–2. B 974 Link Errors

INV R/D	R/D = <result descriptor>
HUB NOT READY	R/D = E000
BUS PARITY ERROR †	R/D = D000
BUS PARITY ERROR †	R/D = D300
DATA TIMEOUT	R/D = C010
HUB PARITY ERROR	R/D = C020
ACCESS DENIED	R/D = C040
HUB TIMEOUT	R/D = C050
PARTNER ERROR	R/D = C060
READ ON WRITE ERROR	R/D = C070
MEMORY PAR ERROR	R/D = C080
WRITE SYSTEM GONE	R/D = C090
EARLY TERMINATION	R/D = C0A0
LENGTH ERROR	R/D = C0B0
CONTROL XFER TIMEOUT	R/D = C0C0
LPC ERROR	R/D = C0D0
AMR WRITE ERROR	R/D = C0E0
MEM BUS PARITY ERROR	R/D = C0F0
DCP DID NOT RESP TO A TEST	

† The HC-2 DLP lacks the "nonstandard" strap

Table 27–3 lists the interface errors.

Table 27–3. Firmware Interface Errors Detected by the MCP

INV R/H ON READ, FUN/OFN	<R/H-type> <orig-function>
INV R/H TYPE, FUNC	<R/H-type>
INV R/H ORIG FUN, O FUNC	<function-number>
INV R/H LSN, LSN	<LSN-value>
INV R/H S-PTR , S-PTR	<S-mem-address>
INV R/H MCS# MCS#	<MCS-number>
NO R/H ORIG FUNC, FUN, OFN	<R/H-type><orig-func>
INV R/H TEXT SIZE, SIZE	<integer>

Section 28
Debug Facility

Overview

The MCP Debug Facility is an interactive, menu-driven tool for debugging the MCP and the user programs at the machine-language level. You can run up to ten Debug sessions at one time.

You initiate a Debug session by entering one of two commands at an ODT. The ODT then becomes an interactive terminal for that Debug session. The Debug Facility enables you to switch back and forth between any of the active Debug sessions and between a Debug session and the normal operation of the ODT.

Initiating a Debug Session

Use the following commands to initiate the Debug Facility:

- The DEBUG command initiates a program and attaches the program to a Debug session.
- The INTERACTIVE DEBUG (ID) command creates a Debug session for a program that is already in the active mix.

When you enter either of these commands, the ODT becomes an interactive terminal for a Debug session. The terminal is reset and internally reprogrammed for the forms mode. A set of user interface menus appears.

When you finish the Debug session, the system reinstates the terminal to its original state and clears all screens.

MCP Initialization

To use the Debug Facility commands, you must include either of the following when you initialize the system:

- The CONTROL DEBUG MCP system configuration record
- The CONTROL DEBUG USER system configuration record in the system configuration file when you initialize the system

The CONTROL DEBUG MCP system configuration record enables you to use the interactive Debug Facility with the MCP or with user programs. The CONTROL DEBUG USER system configuration record enables you to use the interactive Debug Facility only with user programs.

For a complete description of system configuration records, refer to Volume 1.

Utility Commands

The following utility commands are useful for running a number of Debug sessions concurrently:

- The ENTER Debug (ED) command returns you to any of ten Debug sessions from the ODT.
- The QUERY Debug (QD) command displays the number of active Debug sessions on the system and their respective session numbers.

These two utility commands are useful because you can leave the Debug Facility and return to normal ODT operations at any time by using the ODT action on the Main Menu (refer to "Main Menu" in this section).

DEBUG Command

The DEBUG command starts a program and simultaneously attaches the ODT to a Debug session for that program. Enter this command only from an ODT. You can run up to ten Debug sessions concurrently.

The syntax for the DEBUG command appears in Figure 28–1.

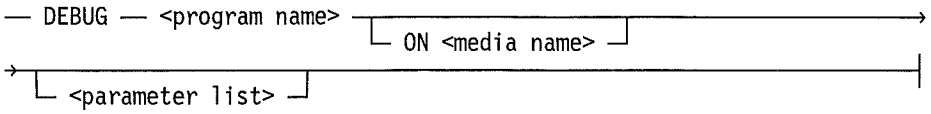


Figure 28–1. DEBUG Command Syntax

<program name>

This option is the name of the program to be attached to an interactive Debug session. The program must contain executable object code and must be stored on the disk or pack specified by the media name.

<media name>

This option refers to the name of the storage device on which the desired program is stored. If you do not specify the media name, the program must reside on the code path.

If you do not include ON <media name>, the code path is searched.

The <media name> can be one of the following:

DISK

This option looks for the desired program on 100-byte media only. If the program is not found, an error message appears on the ODT.

PACK

This option looks for the desired program on all of the system unrestricted packs. If the program is not found, an error message appears on the ODT. The program cannot be found unless it is stored in 180-byte code file format.

To ensure that the program is stored in the correct format, copy it onto the disk pack using the PCOPY utility, or compile the program using the code file media ID option of the COMPILE (Compile Program) command. Refer to Volume 2 for more information.

<pack family name>

This option looks for the desired program on the disk pack family that you specify. If the program is not found, an error message appears on the ODT.

If the program is not found on disk or on unrestricted disk pack, an error message appears on the ODT.

<parameter list>

This option can include up to three parameters that you enter in a program when you begin the program.

These parameters enable you to enter Boolean, string, and integer values that the program can use when you execute the program.

The syntax for the parameter list appears in Figure 28–2.

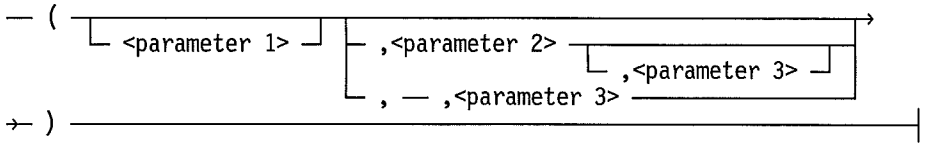


Figure 28–2. Syntax for Parameter List

The following list shows the integer parameters:

- These parameters contain unsigned numeric data up to eight digits in length (8 UN).
- These parameters have values ranging from 0–F.
- These parameters are right-justified and zero-filled in the high-order digits if they are less than the maximum length.

- These parameters cause a syntax error if they are greater than the maximum length.

String parameters:

- These parameters contain alphanumeric data up to six characters in length (6 UA).
- These parameters must be within quotes. To pass a quote as a parameter, a double quote must appear in the parameter string; as in the following example:

```
DEBUG MYPROG (,"TRY""XX").
```

- These parameters can be composed of any EBCDIC characters.
- These parameters are left-justified and blank-filled in the low-order bytes if they are less than the maximum length.
- These parameters will be truncated in the low-order bytes if they are greater than the maximum length.

<parameter 1>

This parameter can be an integer up to eight digits in length. This parameter sets switches at program address 0; as in the following example:

```
DEBUG MYPROG (10100000).
```

<parameter 2>
<parameter 3>

These parameters can be either 8-digit integers or 6-character strings.

Parameter 2 is inserted into the code file at base-relative address 8. Parameter 3 is inserted into the code file at base-relative address 20, as in the following examples:

```
DEBUG MYPROG (,12345678)

DEBUG MYPROG (101,"TOTEST").
```

When either parameter 2 or parameter 3 are specified, you must have either a value or a comma in the preceding parameters, as in the following example:

```
DEBUG MYPROG (,,"TAPE").
```

The program permits a null entry, as in the following example:

```
DEBUG MYPROG (,0,"ABC")
```

This example does not change the switches, but it does insert 00000000 at address 8 and the string ABC at address 20.

Note: *The parameter list replaces slash parameters which were standard on earlier versions of the MCP. The MCP still supports slash parameters.*

Examples

```
DEBUG PAYRL1
```

In this example, the Debug Facility looks for a program named PAYRL1 on the disk named CODEPACK or on unrestricted disk pack. Then it starts the program and attaches it to an interactive Debug session.

```
DEBUG UPDATE ON PACK
```

In this example, the Debug Facility looks for a program named UPDATE on all unrestricted disk packs, starts the program, and attaches it to an interactive Debug session.

```
DEBUG STAT01 ON ACCTS ; FILE OUTPUT = LINEPR/ST1985
```

In this example, the Debug Facility looks for a program named STAT01 on the disk pack family named ACCTS, starts the program, and directs the file named OUTPUT to a specific printer.

INTERACTIVE DEBUG Command (ID)

Use the INTERACTIVE DEBUG (ID) command to debug a task in the active mix. When you enter the ID command from the ODT to invoke a Debug session, the Debug Facility attaches the task corresponding to the task number to the session.

Use this command to debug a task when you do not know the status of the execution, such as in a deadlock or an infinite loop, or during the execution of an independent runner.

The syntax for the ID command appears in Figure 28–3.

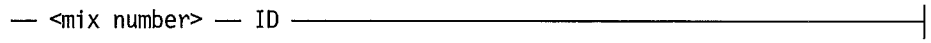


Figure 28–3. ID Command Syntax

<mix number>

This option refers to the mix number of an active job in the mix.

Example

```
55 ID
```

In this example, the syntax invokes the Debug Facility for the program associated with mix number 055.

ENTER DEBUG Command (ED)

The ENTER DEBUG (ED) command enables you to go back to a specified Debug session from the ODT.

The syntax for the ED command appears in Figure 28–4.

Figure 28–4. ED Command Syntax

You can go back to any Debug session from the ODT as long as the Debug session was initiated from that ODT. The session number must follow the keyword ED because it identifies an independent runner that controls the session (as opposed to a normal task number).

The four-digit session number is prominently displayed within each session. You can use the QD command described in this section to view the session numbers of all active Debug sessions.

Example

```
ED 1387
```

In this example, you enter Debug session number 1387 so that you can go back to it.

QUERY DEBUG Command (QD)

The QUERY DEBUG (QD) command identifies the Debug sessions present on the system and displays the four-digit session number of each. You must enter this command from the ODT.

The syntax for this command appears in Figure 28–5.

Figure 28–5. QD Command Syntax

User Interface Menus

The Debug Facility contains several menus as well as an online help facility. These menus are the following:

- Main Menu

  The Main Menu appears after a DEBUG or ID command when the task can be run.

- Status Menu

  If the task cannot be run from the Main Menu, the Status Menu appears.

- Trace Menu

  The Trace Menu enables you to trace and record the execution of the task up to a specified point.

- Breakpoint Menu

  The Breakpoint Menu enables you to set breakpoints in the instructional stream where you want execution of the program to stop.

- State Menu

  The State Menu gives you access to memory so that you can examine and, if you wish, alter memory contents.

Figure 28–6 illustrates the series of user interface menus.

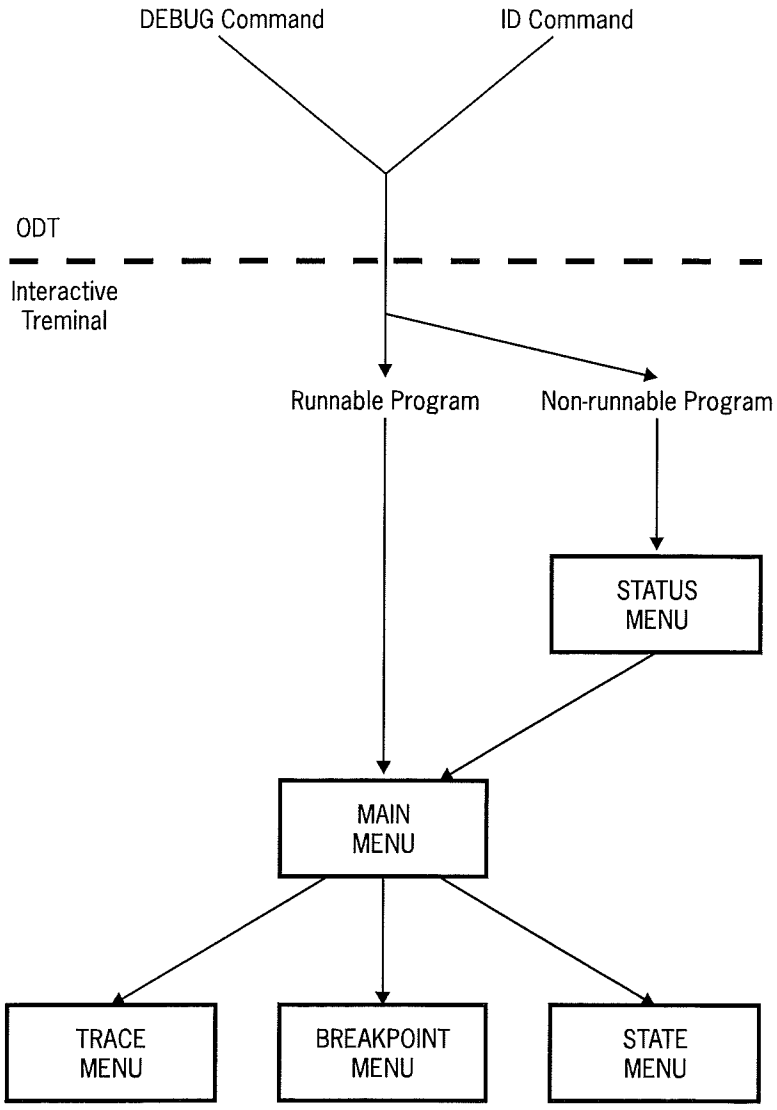


Figure 28–6. User Interface Menus

The State of a Debugged Task

All menus display the state of the debugged task on the second line, as shown in Figure 28–7.

```
-------------------------------------------------------------------------
 TASK MODE STATE   ENVMNT  NIA     COMS      H/A/O/V/T          T/P/M/D/U

-------------------------------------------------------------------------
```

Figure 28–7. Status Line—Example

The following information is included:

TASK	The task number assigned to the task.
MODE	The Debug mode (USR, MCP, SYS, or REL). Refer to the Mode option under "Main Menu" later in this section.
STATE	The internal state of the task: DEBUG—presently debugging STOP—stopped GO ON—running BRK-PNT—Breakpoint Menu invoked HCL-BPT—Hypercall/BCT breakpointed ADDR-BPT—address breakpointed OP-BPT—Opcode breakpointed OVF-BPT—overflow breakpointed BR-BPT—taken branch breakpointed
ENVMNT	The current environment of the task. This field allows six digits; the last three digits are the module number.
NIA	The next instruction address to be executed.
COMS	The comparison toggles. Refer to "State Menu" later in this section.
H/A/O/V/T	An asterisk indicates which breakpoint has been used to stop the program: H—Hyper Call/BCT A—address O—opcode V—overflow T—taken branch Refer to "Breakpoint Menu" later in this section.
T/P/M/D/U	An asterisk indicates which trace action has been set: T—trace P—path M—monitor D—module U—user Refer to "Trace Menu" later in this section.

Main Menu

The Main Menu (refer to Figure 28–8) appears when you enter a DEBUG or ID command, but only when the task to be debugged can be run.

If the task is initially in a waiting state, such as when it is waiting on a resource, the Status Menu appears (refer to Figure 28–9). The Main Menu appears when the task can be run.

From the Main Menu, you can access the Trace, Breakpoint, and State menus for the purpose of executing code in an observable, controlled manner.

```
------------------------------------------------------------------------
  TASK MODE STATE   ENVMNT  NIA     COMS      H/A/O/V/T          T/P/M/D/U
------------------------------------------------------------------------

CMNDS:REF  [ ] STOP [ ] SI [ ] RUN [0000] GO[ ] IGN [  ] QUIT [ ] END [ ]
       :PEEK [      ][   ][  ][      ][    ] POKE[ ] RLE[      ] MODE [   ]
 OTHER MENUS: HELP [ ] TRACE [ ] BREAKPOINTS [ ] STATE [ ] ODT [ ]
```

Figure 28–8. Main Menu

Main Menu Actions

You can invoke the following actions from the Main Menu.

REF — Refreshes the screen by redisplaying the current menu.

STOP — Causes the task to stop executing instructions.

SI — Permits single instruction execution.

RUN — Permits execution of a specified number of instructions before stopping again.

The syntax for the RUN action is:

```
RUN <n>
```

where n is a 4-digit number of instructions.

GO — Permits execution until either an error condition arises or a breakpoint is reached.

IGN — Permits the reported error to be ignored when the task is armed or when the task receives an asynchronous message.

QUIT — Detaches the task from the session and ends the session. The task continues running.

Note: *Use QUIT to terminate a session where a system independent runner is being debugged; QUIT ensures that the task will be detached.*

END — Terminates both the task and the Debug session.

Note: *Do not use END to end the session when debugging a system independent runner; use the QUIT action.*

PEEK — Displays memory at the indicated offset and memory area for the specified length. The syntax for the PEEK action is:

```
PEEK <offset>, <length>, <memory area>, <environment number>,
<task number>
```

The environment and task numbers are optional. The default values
are the current environment and task numbers of the task you are debugging. The digit lengths of these parameters are the following:

Offset—6 digits
Length—3 digits
Memory area—2 digits
Environment number—6 digits
Task number—4 digits

POKE — Enables you to change memory by inserting a P at the beginning of any lines in the display area.

RLE — Displays the Reinstate List Entry of the specified task number.

MODE — Changes the Debug mode to any of the following modes:

USR—Enables Debug for the user program portions of the task only.
MCP—Enables Debug for the MCP portion of the task only.
SYS—Enables Debug for both the MCP and USER portions of the task
REL—Disables Debug completely until either the task terminates or the program reaches an error.

For those portions of the task where Debug is disabled, the task will run at the normal system speed. Enter all actions that resume execution of the task or terminate the task only when the debugged task is in a stopped state. Judicious use of these modes can reduce debugging time significantly. For example, if you want to debug a user program, you need not fault after each instruction executed within the MCP code. Use of the USR mode would prevent this.

HELP — Displays a series of help menus.

TRACE	Displays the Trace Menu (described in this section).
BREAKPOINTS	Displays the Breakpoint Menu (described in this section).
STATE	Displays the State Menu (described in this section).
ODT	Enables you to switch to the ODT. Use the ED command to return to the Debug session from the ODT.

Status Menu

The Status Menu (refer to Figure 28–9) appears when you enter a DEBUG or ID command and the task that you want to debug cannot be run, because it is in a waiting state (for example, when the program is waiting on a resource).

When you can run the task, the Main Menu (refer to Figure 28–8) appears.

```
-------------------------------------------------------------------------
 TASK MODE STATE   ENVMNT  NIA     COMS   H/A/O/V/T   T/P/M/D/U

-------------------------------------------------------------------------
 COMMANDS:REF   [ ] END [ ]
         :PEEK  [      ][   ][  ][      ][    ] POKE[ ] RLE[   ] MODE[    ]
 OTHER MENUS: HELP [ ]
```

Figure 28–9. Status Menu

Status Menu Actions

You can invoke the following actions from the Status Menu.

REF	Refreshes the screen by redisplaying the current menu.
END	Terminates both the task and the Debug session.

PEEK — Displays memory at the indicated offset and memory area for the specified length.

The syntax for the PEEK action is the following:

```
PEEK <offset>, <length>, <memory area>, <environment number>, <task number>
```

The environment and task numbers are optional. The default values are the current environment and task numbers of the task you are debugging. The digit lengths of these parameters are the following:

Offset—6 digits
Length—3 digits
Memory area—2 digits
Environment number—6 digits
Task number—4 digits

POKE — Enables you to change memory by inserting a P at the beginning of any lines in the display area.

RLE — Displays the Reinstate List Entry of the specified task number.

MODE — Changes the Debug mode to any of the following modes:

USR—enables Debug for the User Program portions of the task only
MCP—enables Debug for the MCP portion of the task only
SYS—enables Debug for both the MCP and USER portions of the task
REL—disables Debug completely until either the task terminates or an error is reached

For those portions of the task where Debug is disabled, the task runs at the normal system speed. You can enter all actions that resume execution of the task and terminate the task only when the debugged task is in a stopped state. Judicious use of these modes can reduce debugging time significantly. For example, if you want to debug a user program, you need not fault after each instruction executed within the MCP code. Use of the USR Mode would prevent this.

HELP — Displays a series of help menus.

Trace Menu

You access the Trace Menu (refer to Figure 28–10) from the Main Menu (refer to Figure 28–8). The Trace Menu enables you to direct a variety of trace output to three types of media simultaneously.

```
------------------------------------------------------------------------------
 TASK MODE STATE   ENVMNT  NIA     COMS      H/A/O/V/T          T/P/M/D/U

------------------------------------------------------------------------------
OTHER MENUS:
       HELP [ ] RETURN [ ]------------ TRACE  PATH  PROC  MODULE  USER
DEVICES    :            ODT [ ]          :[ ]   [ ]   [ ]   [ ]    [ ]
                        PRN [    ] / [ ]:[ ]   [ ]   [ ]   [ ]    [ ]
USER OPS [          ]   PBD [ ] /PBT [ ]:[ ]   [ ]   [ ]   [ ]    [ ]
```

Figure 28–10. Trace Menu

Trace Menu Actions

You can invoke the following actions from the Trace Menu.

HELP	Displays a series of help menus.
RETURN	Enables you to return to the Main Menu.

The following table shows the types of trace actions.

TRACE	Traces all instructions executed until the task terminates, an error occurs, or a breakpoint is reached.
PATH	Produces a trace line when instruction execution ceases to be sequential, as with Taken Branch instructions and Enter.
PROC	Produces output lines whenever the task enters or exits a procedure.
MODULE	Produces output lines whenever the task enters or exits a procedure and the environment changes
USER	Produces output lines whenever the task executes a user-selected opcode.

Output (Device) Options

You can output any combination of the trace actions to all devices. The following table shows the output options.

ODT	Enter any non-blank character.
PRINTER	Enter <channel number>/<unit number>. The <channel number> is a 4-digit number and the <unit number> is a 2-digit number.
PBD or PBT	Enter any non-blank character for printer backup disk (PBD) or printer backup tape (PBT). Only one printer backup file can be opened at a time.

User-Selected Opcodes

The USER OPS field enables you to specify a maximum of ten 2-hex-character opcodes.

Breakpoint Menu

You access the Breakpoint Menu (refer to Figure 28–11) from the Main Menu (refer to Figure 28–8). This menu enables you to set a maximum of 32 breakpoints on 10 Hypercall/BCT instructions, 10 instruction addresses, 10 opcodes, or an overflow or taken branch. You can easily set or reset all breakpoints by entering any nonblank character in the able/disable field that precedes each breakpoint.

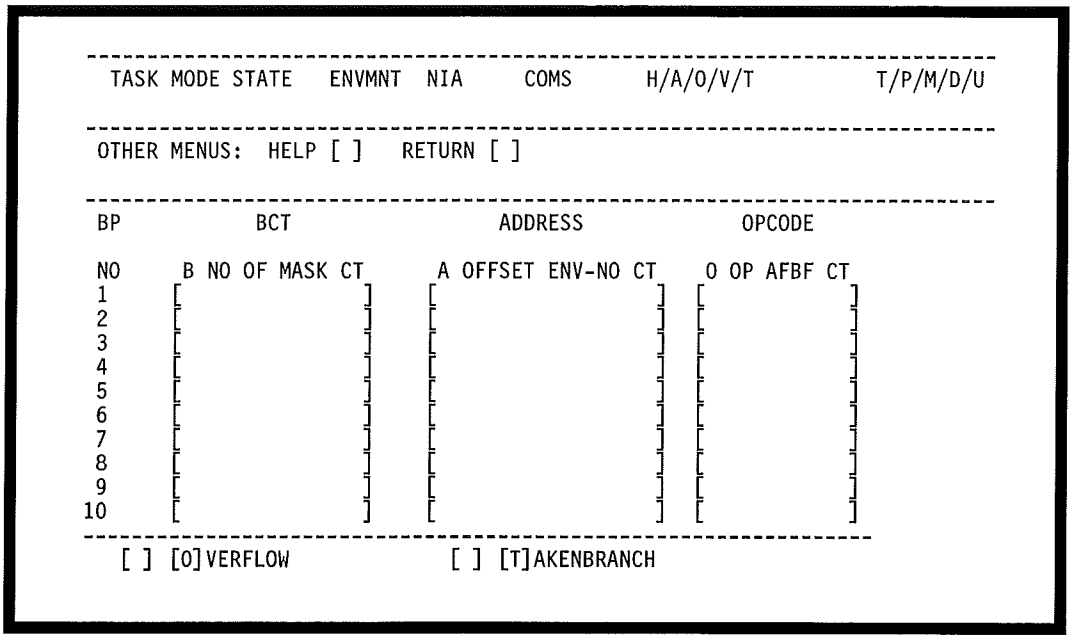


Figure 28–11. Breakpoint Menu

Breakpoint Menu Actions

You can use the following actions on the Breakpoint Menu:

HELP	Displays a series of help menus.
RETURN	Enables you to return to the Main Menu.

Hypercall/BCT Breakpoints

The actions for setting this breakpoint and their default values are shown in Table 28–1.

Table 28–1. Hypercall/BCT Breakpoints

Field	Description	Default
B	Any character	Not applicable
NO	4 digit number	0000
OF	2 digit number	00
MASK	6 hex value	F0F0F0
CT	2 digit number	00

Enter any character in the first field (B) to enable this breakpoint.

The second field (NO) assigns a four-digit number to this breakpoint. For example, to stop on BCT 0214, you enter 0214 (refer to the examples in this section).

The offset (OF) is the number of digits relative to the start of the HCL/BCT parameters.

The mask (MASK) is compared with the offset value, and the task is stopped on a perfect match.

The count (CT) specifies the number of occurrences of the Hypercall/BCT to be skipped before the task will stop.

Address Breakpoint

The actions and their default values for setting this breakpoint appear in Table 28–2.

Table 28–2. Address Breakpoints

Field	Description	Default
A	Any character	Not applicable
OFFSET	6 digit number	000000
ENV-NO	6 digit number	000000
CT	2 digit number	00

Enter any character in the first field (A) to enable this breakpoint.

The address offset (OFFSET) is the instruction address on which the task is to be stopped.

The Environment Table number (ENV-NO) is the MAT number.

The count (CT) specifies the number of occurrences of the address that are to be skipped before the task will stop.

Opcode Breakpoint

This breakpoint stops execution at a given number of occurrences of a specified instruction. The actions and their default values for setting this breakpoint appear in Table 28–3.

Table 28–3. Opcode Breakpoints

Field	Description	Default
O	Any character	Not applicable
OP	3 character string	3 blanks
AFBF	4 digit number	FFFF
CT	2 digit number	00

Enter any character in the first field (O) to enable this breakpoint.

The symbolic opcode (OP) is the mnemonic for the opcode.

The AFBF field is the actual value that is compared to the code stream before the task is stopped.

The count specifies the number of occurrences of the opcode ignored before the task is stopped.

Overflow Breakpoint

You can stop the task when an executed instruction causes the Overflow Toggle to be set.

Taken Branch Breakpoint

This breakpoint enables you to stop the task when the sequential flow of the instruction stream has been broken.

State Menu

You can display and modify the state of the task and the data structure in memory with the State Menu (refer to Figure 28–12). You access the State Menu from the Main Menu (refer to Figure 28–8).

```
-------------------------------------------------------------------------
  TASK MODE STATE   ENVMNT  NIA     COMS      H/A/O/V/T          T/P/M/D/U

-------------------------------------------------------------------------
OTHER MENUS:HELP [ ]  RETURN [ ]-----------------------------------------
  STK [ ]     NSTK [ ]  PSTK [ ]   NIA [    ] COMS [ ] INT [ ] MOD [ ]
  MOP [     ] ACC  [                      ]  IX1 [       ] IX2 [      ]
  IX3 [     ] IX4 [       ] IX5 [      ]    IX6 [       ] IX7 [      ]
```

Figure 28–12. State Menu

State Menu Actions

You can invoke the following actions from the State Menu.

HELP — Displays a series of help menus.

RETURN — Enables you to return to the Main Menu.

STK — Displays the topmost stack frame.

NSTK — Displays the next stack frame relative to the last one displayed.

PSTK — Displays the previous stack frame relative to the last one displayed.

NIA — Displays/modifies the next instruction address.

COMS — Displays/modifies the comparison toggles for the task. The encodings are as follows:

O—Overflow is set–COM 04
E—COMS programs are set (EQUALCOM 03)
L—COMS programs are set (LOWCOM 02)
H—COMS programs are set (IGHCOM 01)
N—COMS programs are set (NULLCOM 01)

INT	Displays/modifies the interrupt mask. The encodings are as follows: O—System overtemperature (INT 20) T—Timer (INT 10) R—Real-Time I/O (INT 04) E—Error I/O (INT 02) N—Normal I/O (INT 01) O—No bits set (INT 00)
MOD	Displays/modifies the mode toggles. The encodings are as follows: F—Soft Fault enabled (MODE 08) P—Privileged task (MODE 04) T—Trace enabled (MODE 02) S—Snap enabled (MODE 01) O—No toggles set (MODE 00)
MOP	Displays/modifies the measurement register.
ACC	Displays/modifies the current accumulator.
IXn	Displays/modifies a system index register, where n is a number between 1 and 7, inclusive.

Debug Session Examples

The following examples show how to use the Debug interactive menus.

Stop User Program

To stop a user program on an instruction address, you mark the BREAKPOINTS field on the Main Menu (refer to Figure 28–13).

```
------------------------------------------------------------------------------
  TASK MODE STATE   ENVMNT  NIA     COMS      H/A/O/V/T          T/P/M/D/U
  0002  SYS DEBUG   D00002  071930   02
------------------------------------------------------------------------------

CMNDS:REF  [ ] STOP [ ] SI [ ] RUN [0000] GO[ ] IGN [  ] QUIT [ ] END [ ]

    :PEEK [000000][000][00][      ][    ] POKE[ ] RLE[000000] MODE [   ]
 OTHER MENUS: HELP [ ] TRACE [ ] BREAKPOINTS [X] STATE [ ] ODT [ ]

------------------------------------------------------------------------------

                    TASK IS ATTACHED AND ENABLED
```

Figure 28–13. Main Menu—Breakpoint Example

When you transmit the Main Menu, the system displays the Breakpoint Menu (refer to Figure 28–14).

```
 TASK MODE STATE   ENVMNT  NIA     COMS      H/A/O/V/T          T/P/M/D/U
 0002  SYS BRK-PNT D00002  071930   02
-------------------------------------------------------------------------
 OTHER MENUS:  HELP [ ]   RETURN [ ]

-------------------------------------------------------------------------
 BP          BCT                ADDRESS            OPCODE

 NO     B NO OF MASK CT      A OFFSET ENV-NO CT    O OP AFBF CT
 1     [               ]    [X 005258 000001   ]  [           ]
 2     [               ]    [                  ]  [           ]
 3     [               ]    [                  ]  [           ]
 4     [               ]    [                  ]  [           ]
 5     [               ]    [                  ]  [           ]
 6     [               ]    [                  ]  [           ]
 7     [               ]    [                  ]  [           ]
 8     [               ]    [                  ]  [           ]
 9     [               ]    [                  ]  [           ]
10     [               ]    [                  ]  [           ]
-------------------------------------------------------------------
   [ ] [O]VERFLOW             [ ] [T]AKENBRANCH
```

Figure 28–14. Breakpoint Menu—Address Breakpoint Example

Transmit this menu and return to the Main Menu (use the RETURN field). The Main Menu appears in Figure 28–15.

```
 TASK MODE STATE   ENVMNT  NIA     COMS      H/A/O/V/T          T/P/M/D/U
 0002  SYS DEBUG   D00002  071930   02          *
-------------------------------------------------------------------------


CMNDS:REF  [ ] STOP [ ] SI [ ] RUN [0000] GO[X] IGN [  ] QUIT [ ] END [ ]

    :PEEK [000000][000][00][      ][    ] POKE[ ] RLE[000000] MODE [   ]
 OTHER MENUS: HELP [ ] TRACE [ ] BREAKPOINTS [ ] STATE [  ] ODT [ ]

-------------------------------------------------------------------------

      ADDRESS BREAKPOINT- TASK STOPPED <BP#01  005258  000001  00>
```

Figure 28–15. Main Menu—Example

Place a mark in the GO field and transmit the menu. The system executes the breakpoint and displays a message showing the breakpoint number, the offset number, the environment number, and the count number.

Using the PEEK and POKE Functions

The PEEK function displays the memory at the indicated address. The POKE function enables you to change the memory displayed.

PEEK Function

On the Main Menu (refer to Figure 28–16), complete the first three fields of the PEEK field (the environment field and task number field are optional). Make sure that the GO field is blank.

```
  TASK MODE STATE   ENVMNT  NIA     COMS      H/A/O/V/T          T/P/M/D/U
  0002  SYS DEBUG   D00002  071930   02
 ------------------------------------------------------------------------


CMNDS:REF  [ ] STOP [ ] SI [ ] RUN [0000] GO[ ] IGN [  ] QUIT [ ] END [ ]
       :PEEK [003292][012][01][      ][    ] POKE[ ] RLE[000000] MODE [   ]
  OTHER MENUS: HELP [ ] TRACE [ ] BREAKPOINTS [ ] STATE [ ] ODT [ ]

 ------------------------------------------------------------------------
```

Figure 28–16. Main Menu—PEEK Input Example

The system displays the contents of the memory at location 003292 for a length of 12 digits (refer to Figure 28–17).

```
  TASK MODE STATE   ENVMNT  NIA     COMS      H/A/O/V/T          T/P/M/D/U
  0002  SYS DEBUG   D00002  071930   02
 ------------------------------------------------------------------------


CMNDS:REF  [ ] STOP [ ] SI [ ] RUN [0000] GO[ ] IGN [  ] QUIT [ ] END [ ]
       :PEEK [003292][012][01][      ][    ] POKE[ ] RLE[000000] MODE [   ]
  OTHER MENUS: HELP [ ] TRACE [ ] BREAKPOINTS [ ] STATE [ ] ODT [ ]

 ------------------------------------------------------------------------
  003292  012  01     F0F0F0F0F0F1
```

Figure 28–17. Main Menu—PEEK Display Example

POKE Function

Use the POKE function to change the instructions in the memory location that appears in Figure 28–17.

Perform the following steps (refer to Figure 28–18):

1. Clear the PEEK fields.
2. Mark the POKE field.
3. Move the cursor to the display of the memory address.
4. Place a P immediately before the address number.
5. Change the memory contents.
6. Transmit.

```
   TASK MODE STATE   ENVMNT  NIA     COMS      H/A/O/V/T          T/P/M/D/U
   0002  SYS DEBUG   D00002  071930   02
  ------------------------------------------------------------------------


 CMNDS:REF  [ ] STOP [ ] SI [ ] RUN [0000] GO[ ] IGN [  ] QUIT [ ] END [ ]

     :PEEK [      ][  ][  ][      ][    ] POKE[X] RLE[000000] MODE [   ]
  OTHER MENUS: HELP [ ] TRACE [ ] BREAKPOINTS [  ] STATE [ ] ODT [ ]

  ------------------------------------------------------------------------


    P003292  012  01     F1F1F1F1F1F1
```

Figure 28–18. Main Menu—POKE Example

Using the Trace Functions

The Trace Menu enables you to direct trace output to a variety of media.

Mark the TRACE field on the Main Menu to access the Trace Menu (refer to Figure 28–19). Mark the desired fields. In this example, the request is for a complete trace of instructions to be sent to a printer backup file on disk (PBD).

When you transmit the menu, the system returns the message:

```
NEW PRINTER BACKUP-FILE HAS BEEN OPENED
```

```
TASK MODE STATE   ENVMNT  NIA     COMS      H/A/O/V/T          T/P/M/D/U
0002  SYS DEBUG   D00002  071930   02                          *
 ------------------------------------------------------------------------

  OTHER MENUS:
        HELP [ ] RETURN [ ]------------ TRACE  PATH  PROC  MODULE  USER
  DEVICES    :           ODT [ ]          :[ ]   [ ]   [ ]    [ ]   [ ]
                         PRN [    ] / [ ]:[ ]   [ ]   [ ]    [ ]   [ ]
  USER OPS [FFFFFFFFFF]  PBD [X] /PBT [ ]:[X]   [ ]   [ ]    [ ]   [ ]
```

Figure 28–19. Trace Menu—Example

Debug Session Errors

When the system is executing on behalf of a debugged task and encounters an error, it reports the error to the session with a hardware interrupt. The exact nature of the fault appears in the fault indicators (refer to Table 28–4).

Table 28–4. Fault Indicators

Digit	Bit	Type of Fault
72	8	Hard Memory Area
72	4	Trace
72	2	Invalid Arithmetic Data
72	1	Soft Memory Area
73	8	Invalid Instruction– More information in the IEX Byte (refer to Table 28–5)
73	4	Uncorrectable Memory Parity Error
73	2	Address Error– More information in the AEX Byte (refer to Table 28–6)
73	1	Instruction Timeout
74	8	Stack Overflow
74	4	Accumulator Trap taken
74	2	Snap Picture taken
74	1	Soft Fault

Table 28–5 lists the additional information generated by the invalid command extensions (digits 80–81).

Table 28–5. Invalid Command Extension (IEX)–Digits 80–81

Description	Value
GENERAL	00
Invalid Operator Code	01
Privileged Mode Violation	02
Invalid Address Controller	03
Stack Overflow	04
Counter Overflow/Underflow	05
Invalid Field Comparison	06
Invalid Operand Field	07
Invalid AF or BF, GENERAL	20
Literal Not Permitted	21
Invalid Literal	22
Invalid Indirect Field Length	23
Invalid Variant	24
Invalid AF	25
Invalid BF	26
Invalid Privileged Primary Access	31
Invalid Privileged Sec. Access	32
Invalid Attempt to Modify Original or Fault Memory Table Entry	35
Copy Protection Violation	36
Stack Protection Violation	37

Table 28–6 lists the additional information generated by the invalid address extensions (digits 78–79).

Table 28–6. Invalid Command Extension (IEX)—Digits 78–79

Description	Value
GENERAL	00
Invalid Address Relationship	01
Hypercall Function Limit Error	02
Odd Operand Address	03
Invalid MAT Entry	04
Index Register, GENERAL	10
Invalid Arithmetic	11
Index Register Contains Undigit	12
Invalid Base Indicant	13
IX3 Negative on RETURN	14
IX3 Odd on RETURN	15
Base/Limit Error, GENERAL	20
Command Fetch	21
Address Resolution	22
Operand Write	23
Operand Read	24
Global Link Address	25
Address Undigit, GENERAL	30
Command Fetch	31
Address Resolution	32
Operand Write	33
Operand Read	34
Global Link Address	35
Branch Address, GENERAL	40
Address >= Limit	41
Address Contains Undigit	42
Odd Address	43
Invalid Environment Descriptor	50
Invalid Environment Number	51
Invalid Most Significant Digit	52
Index Contains Undigit	53
Memory Area No. Contains Undigit	54
Environment Number or Memory Area Number Out of Range,	56
GENERAL	
Environment No. Out of Range	57
Memory Area No. Out of Range	58

continued

Table 28-6. Invalid Command Extension (IEX)—Digits 78-79 (cont.)

Description	Value
Invalid Memory Area Table Entry	60
Invalid Environment Number Invalid Most Significant Digit Index Contains Undigit Memory Area No. Contains Undigit Environment Number or Memory Area Number Out of Range,	61 62 63 64 66
GENERAL Environment No. Out of Range Memory Area No. Out of Range	 67 68

Error Messages Related to Debug Commands

The following paragraphs describe the error messages that the DEBUG, ID, and ED commands can generate.

`EXCEEDED DEBUG SESSION LIMIT`

There are ten active Debug sessions on the system. No new sessions can be invoked until at least one of the current sessions terminates. Use the QD command to identify the current sessions.

`ERROR IN INITIATING DEBUG SESSION`

There was an internal error in the session.

`TASK CURRENTLY BELONGS TO ANOTHER SESSION`

The task that you attempted to attach to this session was being debugged in another session.

`DEBUG SESSION DOES NOT BELONG TO OCS`

The session number does not correspond with the ODT from which this command was issued.

`DEBUG SESSION UNASSIGNED`

The session number is invalid and does not correspond to any current session.

`FOUR DIGIT SESSION NO REQD`

You must include a four-digit session number in the command.

Section 29
QWIK Disk

Overview

The V Series QWIK Disk utilizes extra system memory to simulate a high performance 100-byte disk subsystem. This section describes the system bottlenecks that QWIK Disk can remedy. It also describes file-selection criteria, file-placement suggestions, installation and cold-start procedures, as well as operational and programming considerations for QWIK Disk.

QWIK Disk is easy to install. You can use it on any V Series system and it is fully compatible with all releases of the V Series MCP/VS. You do not need to change any existing software to use QWIK Disk.

QWIK Disk is very easy to use because it simulates existing 100-byte disk subsystems. You can read and write to files in QWIK Disk in the same way that you read and write files to any other 100-byte disk. To obtain the maximum performance from your system, you must be selective when you decide which files to put in QWIK Disk. Because of its limited capacity, QWIK Disk is not recommended for use as default disk.

Caution

Because QWIK Disk makes use of main memory, information stored in QWIK Disk can be lost because of power failures or because of some system faults. Critical files should be considered for QWIK Disk only if you have adequate backup procedures or if the performance benefits justify the potential recovery overhead.

QWIK Disk is an excellent storage medium for files that have a high I/O volume. It is also an excellent medium for read-only files in a transaction processing system that requires a short response time. When you select files for QWIK Disk, keep in mind a basic size-to-I/O ratio. Use the smallest files with the highest I/O volume.

QWIK Disk significantly reduces the amount of time it takes to complete an I/O; it does not reduce the number of I/Os that take place. Instead, it speeds them up in one of the following ways.

- By using RAM memory for disk storage, QWIK Disk eliminates "seek time," which is the time spent waiting as a disk head goes through the mechanical process of finding a sector and reading data.
- QWIK Disk has a high rate of data transfer.

Installing QWIK Disk

To install QWIK Disk, you must do the following:

1. Load the latest IOP firmware from floppy disk.
2. Determine the amount of the memory the MCP will use and the amount QWIK Disk will use.
3. Use two maintenance processor commands to set the MCP limit and the QWIK Disk option.
4. Cold-start the system with two special records in the system configuration file.

One of the special records tells the system how much storage QWIK Disk has. The other special record reserves channel 40 for QWIK Disk.

For information on initializing your system, refer to "Maintenance Processor Commands" in this section, "Initializing the System" in the Volume 1 and in the hardware documentation for your processor.

When the installation and cold-start are complete, you can read and write files to and from QWIK Disk as you would to and from any other disk subsystems.

Figure 29–1 shows an example of the division of memory between the MCP and QWIK Disk.

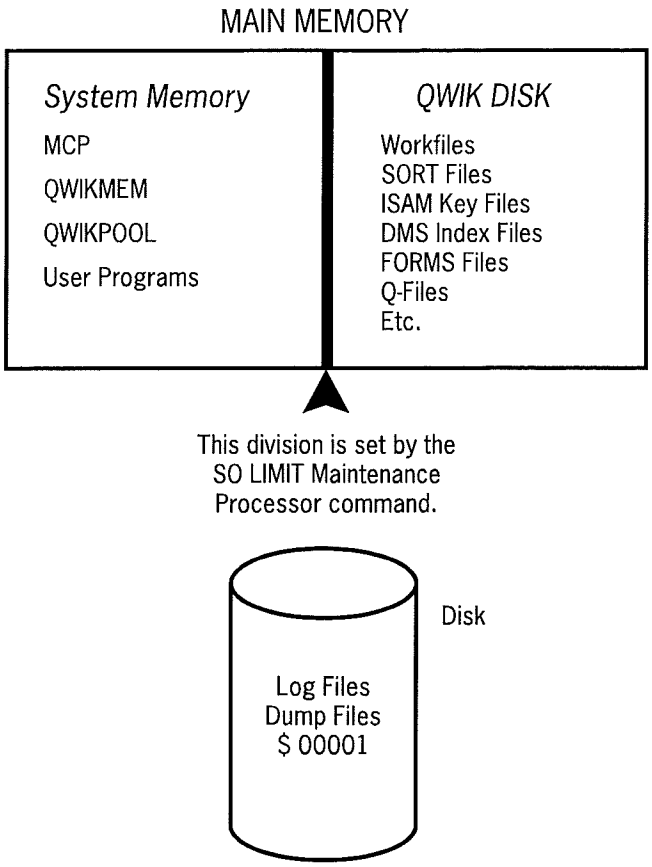


Figure 29–1. Example Memory Partition between the MCP and QWIK Disk

Performance Improvement Guidelines

The performance figures for QWIK Disk vary depending on the file classes and individual files you select for QWIK Disk. Heavy peripheral activity can cause performance variance.

The following paragraphs describe performance guidelines that are based on benchmark tests run in empty environments. For consistency, the performance improvements are expressed as *factors* computed as according to the following formula:

```
Factor = Old Time/New Time
```

QWIK Disk Peripheral Performance

Because QWIK Disk eliminates seek time, the average QWIK Disk I/O requires only 1 to 2.4 milliseconds. For QWIK Disk, I/O time is all data transfer time. QWIK Disk can transfer the largest possible block size (20,000 bytes) in 16 milliseconds.

QWIK Disk I/O speed can be at least 9 times faster than devices with movable heads, such as LAK 206 and 207.

General System Performance

When you use QWIK Disk in a general production or development environment, the performance results vary significantly depending upon the files you place in QWIK Disk. QWIK Disk improvement usually varies directly with the percentage of total I/Os shifted to QWIK Disk.

Code File Overlays

Most systems use the QWIKPOOL option to avoid disk-overlay accesses. Unisys recommends that you do overlays in the QWIKPOOL.

Sort Performance

Sort and SRTUTL performance can improve based on placement of input, output, and/or work files in QWIK Disk. SORT: shows marginal improvement with work files in QWIK Disk.

SORT input and output files usually receive more I/Os than work files (which generally have large blocking). For optimum SORT performance, place at least the SORT input and work files in QWIK Disk.

DMSII and ISAM Performance

Because of the large number of files involved, DMSII and ISAM performance figures vary significantly. Because of their small size, ISAM key files and DMSII index files (sets and subsets) are good candidates for QWIK Disk.

Functions with high I/O waiting time, such as ISAM, DMSII DELETE, and OUTPUT WRITE (CREATE/STORE) operations are particularly effective if the files involved reside in QWIK Disk.

Transaction System Performance

Transaction-based mixes generally consist of a message control system (MCS), transaction-processing programs, and a Data Management System II (DMSII) Data Base Program (DBP). DMSII structures will usually receive the highest I/O traffic.

System Analysis–An Overview

The following paragraphs outline a method of analysis that can help you to isolate the types of system bottlenecks that QWIK Disk can remedy. These paragraphs include examples of using QWIK Disk in different types of environments and a list of the various system reports and tools that can help you with this analysis.

Because Unisys systems are used worldwide for many different applications, this analysis can be considered only an outline to complement your understanding of your system environment.

To integrate QWIK Disk into your processing environment, you need to study the following system characteristics to isolate the types of bottlenecks QWIK Disk can remedy:

- System use
- Memory use
- Peripheral activity

System Use

Examine the use of the system to determine the amount of MCP idle time. If the MCP idles more than 10 percent, an opportunity for QWIK Disk exists, and you should proceed with the rest of the analysis. If not, the MCP is *system bound*, and QWIK Disk can do little to improve its total performance.

Memory Use

After you examine the system use and find out the MCP idle time, look at the system memory use to improve your MCP and program overlay performance.

Find out if the MCP spends more than 10 percent of its waiting time waiting for MCP or program overlays. If so, you might be able to improve your system performance by moving the MCP code file or programs to QWIK Disk.

Remember, QWIKPOOL is the best tool to use for overlays. If it is not already in use, it can increase your system overlay performance more than QWIK Disk.

However, QWIK Disk can improve the performance of programs with overlays too big for the QWIKPOOL. Oversized overlays normally come from disk. But if you move the program into QWIK Disk, they will go to RAM memory instead.

Peripheral Activity

After you improve your system memory performance, you can begin to isolate its I/O bottlenecks.

Determine which peripheral devices have the most I/O traffic. Find out if traffic is spread out evenly among all the peripheral devices or if it is unevenly distributed. If it is uneven, you can improve performance by relocating files onto different peripheral devices as well as onto QWIK Disk.

Also look for high I/O traffic to classes of files, individual files, and individual file areas. In these cases, move the groups of files, the individual file, or the individual file areas to QWIK Disk.

Remember to move the smallest files with the highest I/O volume to QWIK Disk.

Performing a System Analysis

The following paragraphs describe the system analysis you need to perform.

You can use both the FLAME software package and the TABSII software package to analyze your system.

Step 1. Examine System Utilization and Determine the MCP Idle Time

Examine your system use to find out the amount of time the MCP spends waiting. If it idles more than 10 percent, an opportunity for QWIK Disk exists and you should proceed with the analysis. If not, the MCP is *system bound*, and QWIK Disk can do little to improve its total performance.

The MCP/USER graphs produced by the FLAME software package can help you determine the MCP idle time. If you do not have FLAME, a "soaker" program will perform a similar job. A soaker program is a priority 1 program that loops, continually counting, soaking up the unused system time when no other program is executing and the MCP is idling.

Figure 29–2 illustrates a system in which the MCP idles (waits) 57 percent of the time. The system spends another 40 percent executing user jobs, and the MCP executes 3 percent of the time.

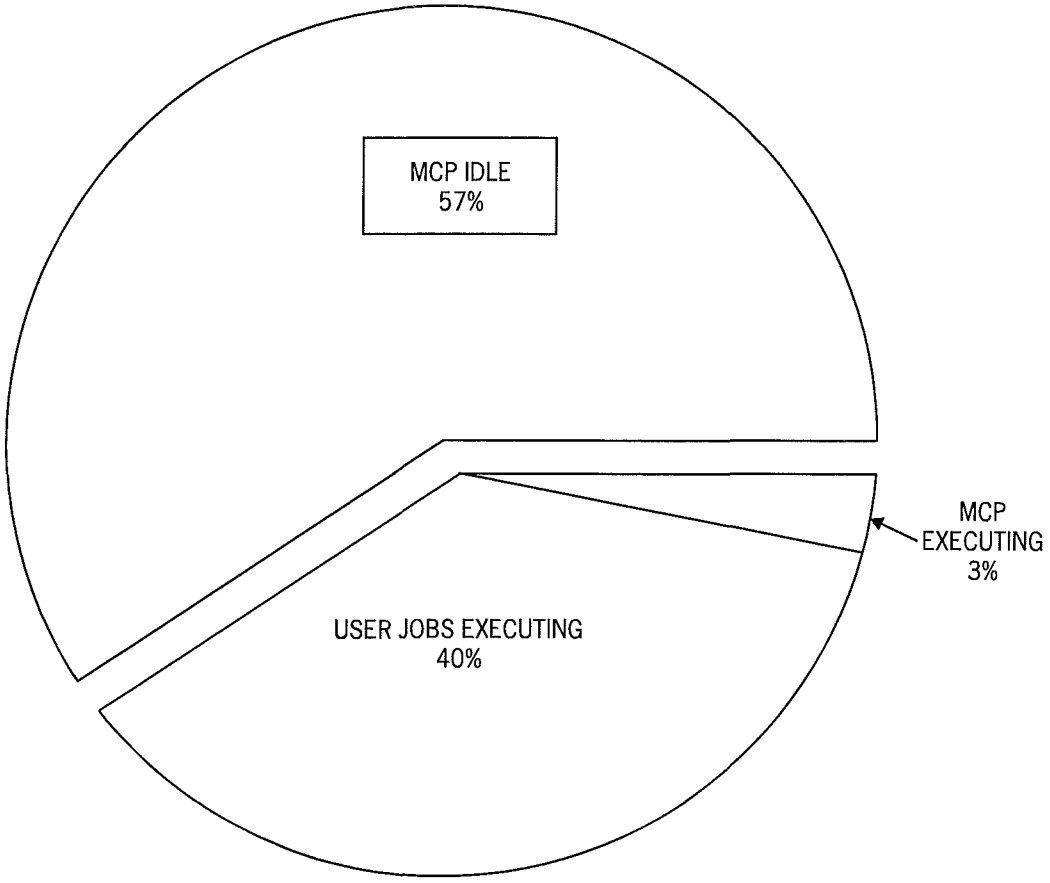


Figure 29–2. Comparison of Idle Time to Execution Time

Step 2. Determine the Time Spent Waiting for MCP and Program Overlays

If your MCP spends more than 10 percent of its time waiting, determine what it waits for.

Figure 29–3 illustrates a number of factors that contribute to MCP waiting time.

The following are overlays for which QWIK Disk can reduce waiting time:

- MCP overlays
- Program overlays

The following are I/Os for which QWIK Disk can reduce waiting time:

- I/Os to disk or disk pack
- I/Os to classes of files
- I/Os to individual files

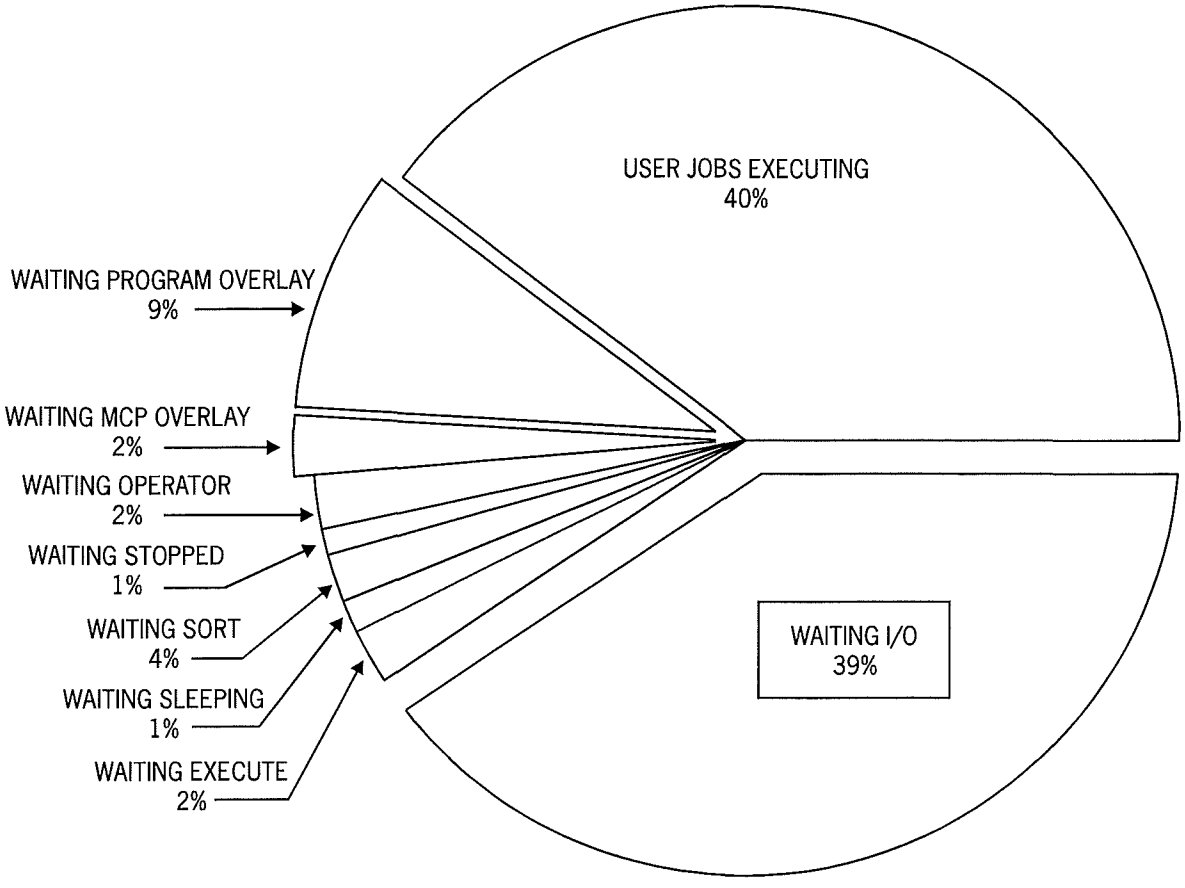


Figure 29–3. Factors That Contribute to MCP Waiting Time

Step 3. Examine MCP and Program Overlays to Determine Memory Utilization

Find out how much time is spent waiting for MCP and program overlays.

MCP Overlays

Use the FLAME graphs in the Memory Analysis series of reports entitled "MCP Overlays" to get an idea of how much the system waits for MCP overlays. Refer to the *V Series FLAME Installation and Operations Reference Manual* for more information.

The QWKMEM option is the primary tool for speeding up MCP overlays. This option provides an area of main memory to be used exclusively for MCP overlays. The most recently called MCP overlay is stored in this area so that it can be retrieved from memory when it is needed, eliminating disk I/O time. Refer to Volume 1 for more information about QWKMEM.

If the system frequently accesses MCP overlays from disk even though you have set the QWKMEM option, you can improve system performance by moving the MCP code and files to QWIK Disk.

Program Overlays

Examine the amount of time programs spend waiting for overlays. The FLAME graphs in the Memory Analysis series of reports entitled "Percent of Jobs In MIX Waiting Program Overlays" and "Number of Program Overlays Called" can help with this analysis. Refer to the *V Series FLAME Installation and Operations Reference Manual* for more information. Also, the TABSII reports "QWIKPOOL Usage Summary" and "Average Overlay Call Hit Rate" are useful. Refer to the *B 2000/B 3000/B 4000/V Series TABSII Installation and Operations Guide* for more information.

If more than 10 percent of waiting time is spent waiting for program overlays, you should be able to improve performance. The primary goal is to increase the amount of memory allocated to the QWIKPOOL, which is an area of memory reserved for the most recently used program overlays. QWIKPOOL reduces the number of overlays that must be read from disk.

Figure 29–4 illustrates the amount of time spent waiting for overlays.

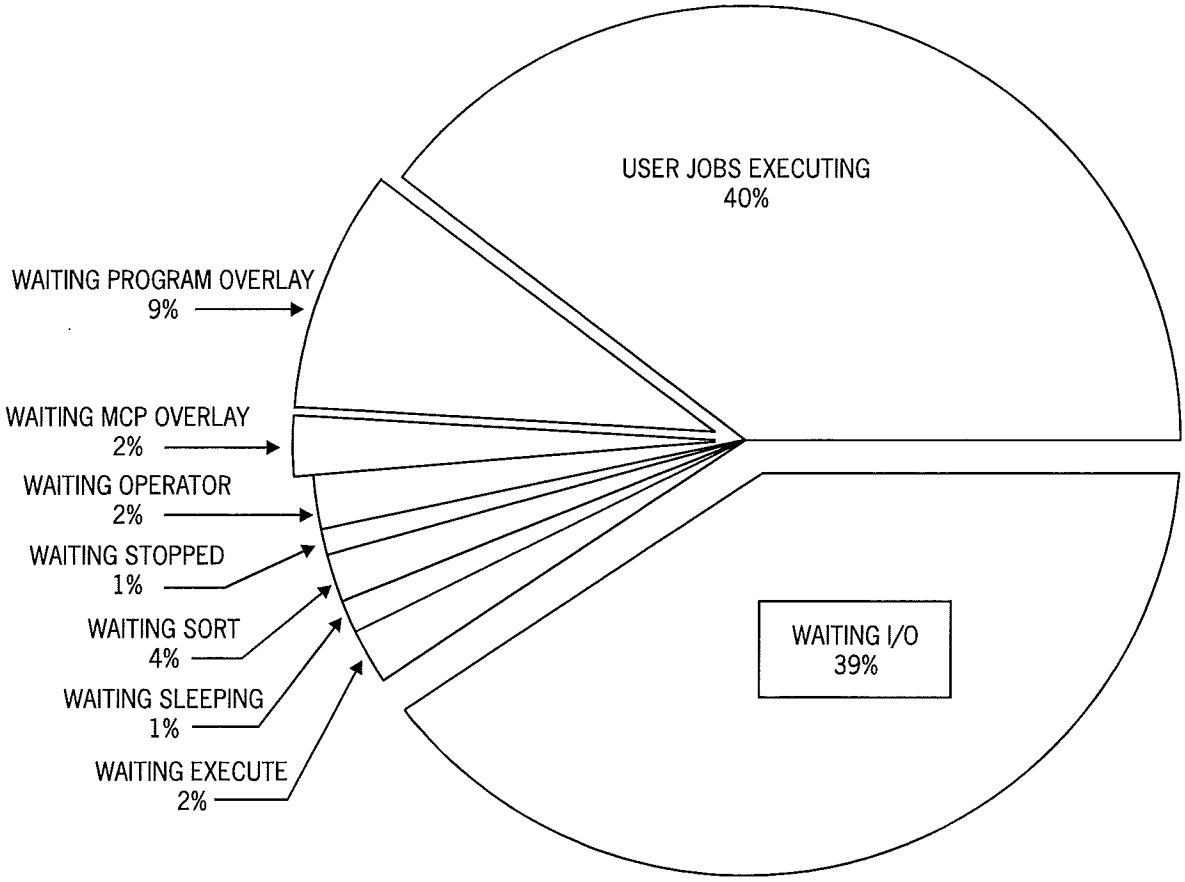


Figure 29–4. Amount of Time the System Spends Waiting Overlays

Step 4. Examine I/O Wait Time for Peripheral Activity

Examine the Run Log (RLOG) I/O counts and the FLAME report entitled "Job Waiting" to find out how much time the MCP spends waiting for I/O. If the MCP spends more than 10 percent of its waiting time waiting for I/O, proceed with the following analysis.

Find out how much time it spends waiting for I/Os to disk or disk pack, MCP code files, various classes of files, and individual files. This information will enable you to fit your system into one of the following categories:

- **System bound.** Jobs are being executed at least 90 percent of the time. QWIK Disk can offer little performance improvement.
- **Memory bound.** The system is waiting for MCP or program overlays more than 10 percent of the time. QWIK Disk, QWKMEM, and QWIKPOOL can improve overlay performance.
- **I/O bound.** A high percentage of waiting time is spent waiting for I/Os to peripheral devices and files. You can improve system performance by moving files with a high I/O volume to QWIK Disk.
- **I/O and memory bound.** A high percentage of the system waiting time is spent waiting overlays and I/Os. First, improve overlay performance by maximizing the QWKMEM and QWIKPOOL; then decide if you can improve the time spent waiting for I/Os. The following steps provide instructions for isolating the files and peripheral devices.

Figure 29–5 illustrates the amount of time that the MCP waits for overlays.

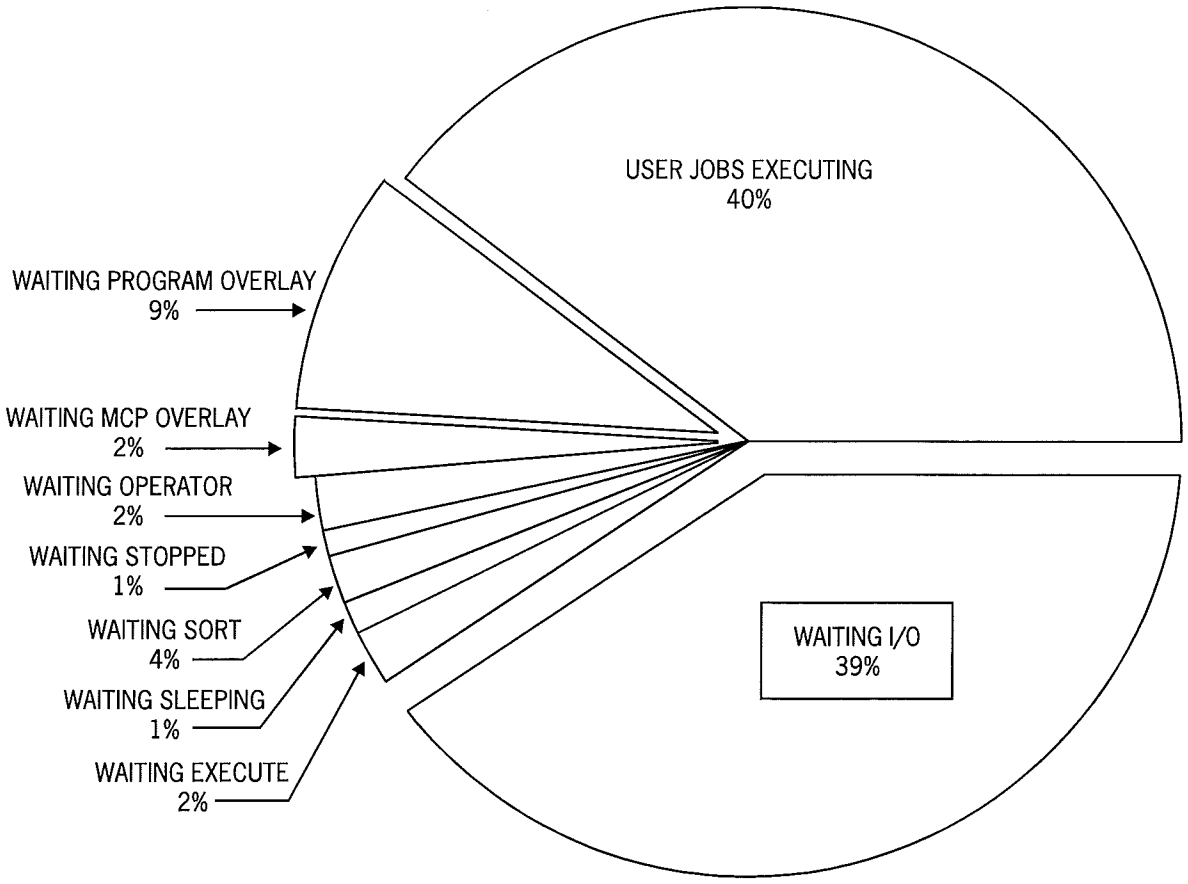


Figure 29–5. Amount of Time the MCP Spends Waiting I/O

Step 5. Determine I/O Counts to Disk and Disk Pack

If a majority of I/Os go to disk pack, your system can be characterized as a disk pack-based system. If the majority are going to disk, it is a disk-based system. Determine the nature of your system and apply the following:

- **Disk Pack Based.** If your system is a disk pack-based system, you can use QWIK Disk as a form of cache by moving files in and out as needed. Doing so may require operational or programming changes. Because these files are moved to QWIK Disk for temporary use, you must remember to copy them back to disk pack when you finish and thus update the original files. The analyses described in steps 6 through 9 can help you identify the types of files or the individual files with the highest I/O volume.
- **Disk Based.** If your system is a disk-based system, proceed with the next analysis.

Figure 29–6 illustrates a system that has 65 percent of its I/Os going to disk and 30 percent going to disk pack. The remaining I/Os go to other media. This system is a disk-based system.

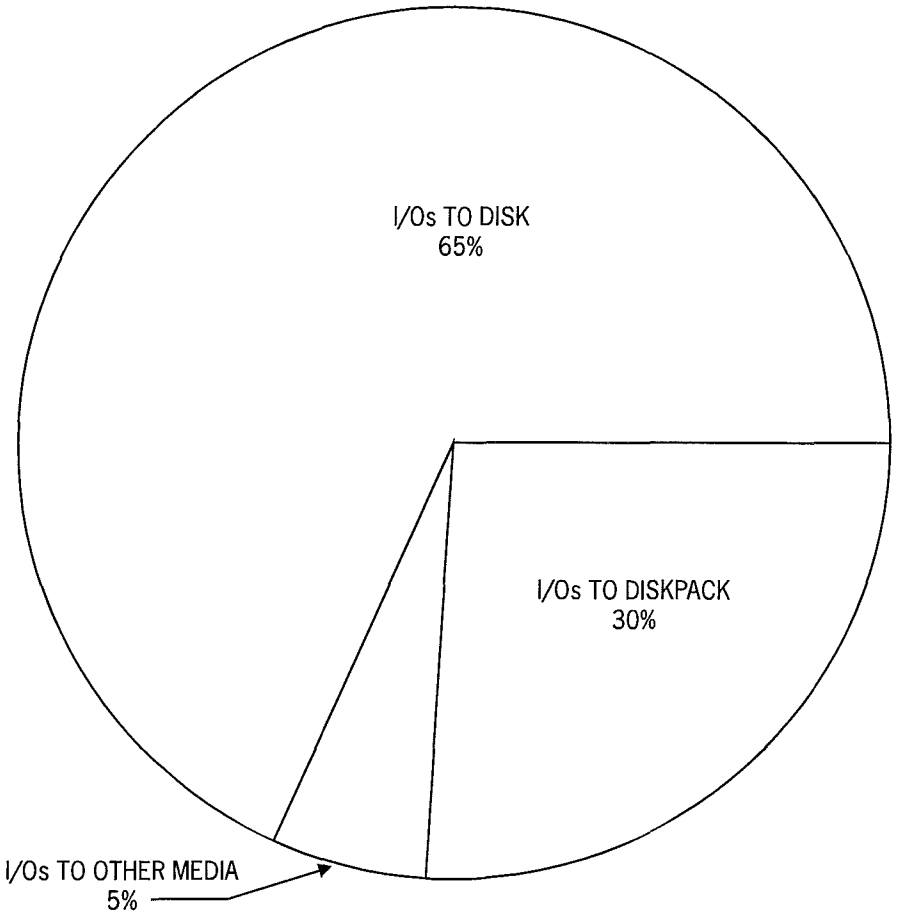


Figure 29–6. Pattern of I/O Traffic to Disk, Pack, and Other Media

Step 6. Analyze General File I/O

If excess MCP waiting time is spent waiting for I/O, use FLAME and Run Log (RLOG) reports to identify the files or file types the MCP waits for most often.

This file I/O analysis is broken down according to the following categories:

- High I/O to MCP files
- High I/O wait time on individual files
- High I/O wait time on various classes of files including
  - Work files
  - Data files
  - Data base files
  - ISAM files

Figure 29–7 illustrates the file types for which a system is waiting.

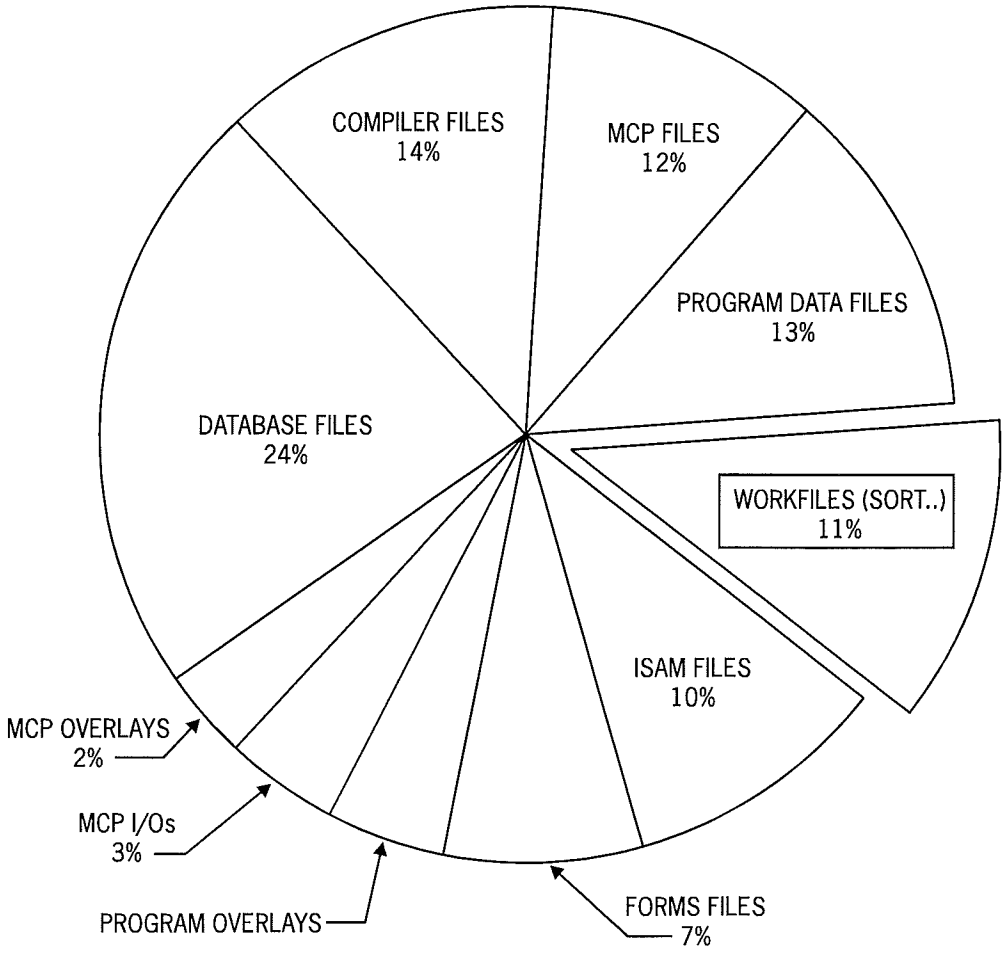


Figure 29–7. Disk I/O Analysis of a System Waiting I/O to Different File Type

Step 7. Examine I/O Volume to MCP Files

Use FLAME graph 37, "MCP I/O," to determine if the MCP files have a high I/O volume.

- If they do, consider moving them to QWIK Disk.
- If not, proceed with the next analysis.

Step 8. Examine I/O Volume to Classes of Files

Use the Run Log (RLOG) to identify the file classes (such as database files) that have the highest I/O volume. Make a list of the worst offenders and consider moving the smallest files with the highest I/O volume to QWIK Disk.

You can route certain classes of files to QWIK Disk either by declaring a work file subsystem when you perform a cold-start or by using such system options as WRKP. Refer to "Files" and "Programming Considerations" later in this section for more information.

Step 9. Examine I/O Volume to Individual Files

Use the Run Log (RLOG) to determine specific file I/O counts and the percentage of total I/O wait time for a job. List the worst offenders and consider moving the smallest files with the highest I/O volume to QWIK Disk. Refer to "General File Selection Criteria" later in this section for information about refining the list.

Figure 29–8 illustrates a disk I/O analysis.

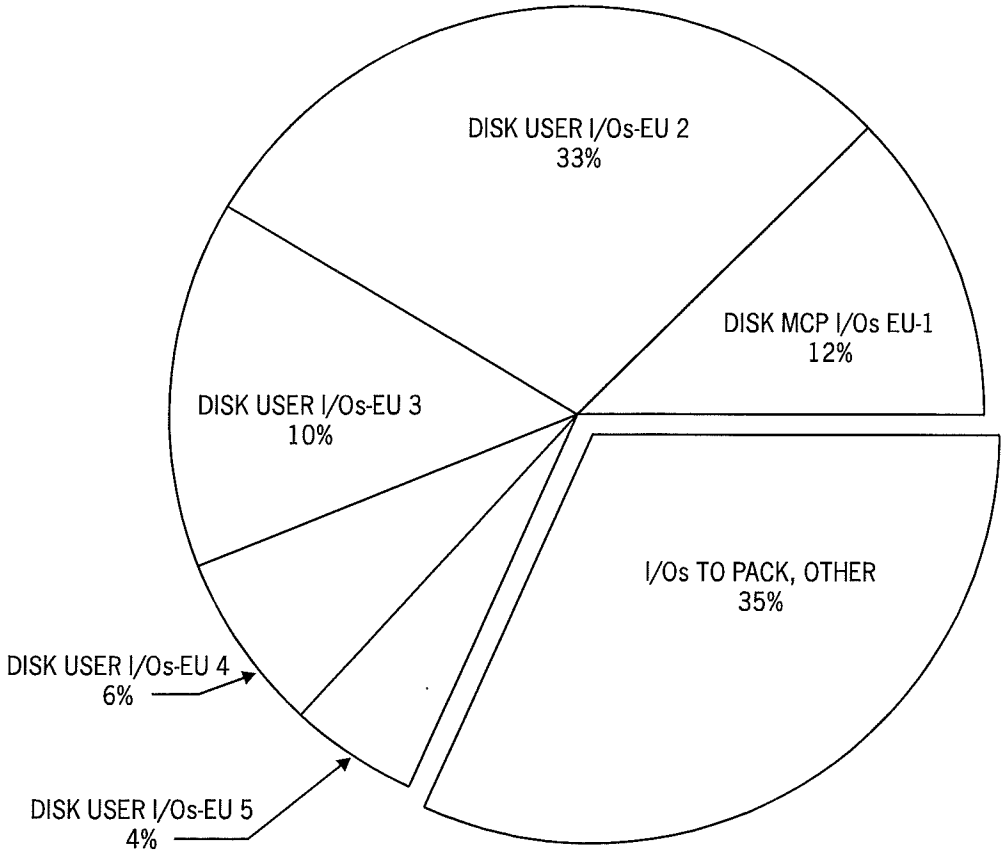


Figure 29–8. Types and Locations of I/Os

Step 10. Analyze Channel/Subsystem/ID

Underconfigured systems and lopsided peripheral configurations can cause system bottlenecks.

If a disk ID or spindle is disproportionately busy, consider moving some of its files to QWIK Disk. Use the FLAME peripheral activity graphs to analyze the disk subsystem and disk ID.

Evenly divide the percentage of time a peripheral is in use over the disk channels of the system. If a disk channel is consistently used more than 40 percent of the time, you might

want to move some of its files to QWIK Disk. Because of its speedy I/O transfer rate, QWIK Disk use will rarely exceed 25 percent.

Selecting Dynamic Profile or Static Profile

Because QWIK Disk is a volatile storage device (it loses its files if the power goes off), you should now determine whether the system would be best served by a static area where you load specific files that reside in QWIK Disk for the entire production cycle or by a constantly changing (dynamic) area.

Static Profile

A static environment with files of permanent residency can still support some dynamic files. However, you should develop a backup procedure for all noninquiry QWIK Disk files.

A static profile contains files with the following characteristics:

- Long file access span
- Inquiry or limited update file
- ISAM
- Code files (for overlays)

Dynamic Profile

A dynamic profile includes files with the following characteristics:

- Short file access span
- Temporary or output
- Work files

For dynamic files, consider the following question: Are the files in QWIK Disk recoverable by a user-coded restart?

Sample Environments

The following paragraphs contain examples of how you might use QWIK Disk in different types of environments, including development, production, and disk pack environments.

Sample of Development Environment

In this sample development environment, both dynamic (steadily changing) files, and static (nonchanging) files are placed in QWIK Disk.

Compilation of dynamic files are frequently cycled through the QWIK Disk to achieve a high compilation rate. Also, heavily accessed work files are routed to QWIK Disk. In

addition, QWIK Disk holds static files, such as an inquiry-only file containing software reference sections.

Sample of Production Environment

In this sample production environment, file-access spans are generally long, and file residency is more nearly permanent. A goal for this type of environment is to mitigate bottleneck structures. Critical issues include online response time and transactions per hour. This environment would be considered a static one where some data update can occur.

Sample of Disk Pack Environment

In this sample disk pack environment, nearly all accessed files reside on 180-byte media. QWIK Disk serves as a type of disk pack cache, where heavily accessed disk pack files are copied to QWIK Disk for use. Then they are copied back to disk pack to update the originals.

Because all QWIK Disk files are backed up on disk pack, you might consider using a QWIK Disk-resident MCP to facilitate MCP overlays and directory I/Os. This option is not recommended unless the impact of directory loss is minimal.

Note: *If the disk directory is lost from QWIK Disk, you must perform a cold-start to recover.*

Measurement Techniques

The following paragraphs discuss the system commands and reports you use to monitor system performance. Use this information to decide which files to put in QWIK Disk. The reports show the sizes of various files and the number of program overlays and peripherals QWIK Disk uses.

The commands include the following:

- KA (reports on file sizes and their location)
- KS (reports all areas of disk use by ID number)
- OL (reports on channel activity)
- WO (reports on overlay activity in the QWIKPOOL)

The utility programs include the following:

- RLGOUT (reports on file sizes and I/O activity)
- FLAME (reports on the overall performance of QWIK Disk)

Using FLAME

FLAME is an effective tool for measuring certain performance areas. You should monitor the system environment both before and after QWIK Disk file placement.

Table 29–1 lists the graphs that should be of particular interest. Refer to the *V Series FLAME Installation and Operations Reference Manual* for more information.

Table 29–1. FLAME Graphs

Graph	Subject
30–37	Jobs in the mix marked waiting
11	Percentage of time user jobs are executing
12	Percentage of time MCP is executing and idling
21	MCP disk overlays
22	MCP main memory overlays
23	Number of program overlays from memory
42	MCP I/Os versus total I/Os to disk
58	Percent of time channels are busy
57	Number of I/Os to channels

Files

The following paragraphs discuss criteria for selecting the files that you want to place in QWIK Disk. The discussion includes the files that are possible candidates for placement in QWIK Disk, and general procedures for loading and unloading files to and from QWIK Disk.

General File Selection Criteria

QWIK Disk is ideal for files that have a high I/O volume or that require a short response time, or both. The performance improvement that you obtain by putting these files in QWIK Disk depends on the ratio of I/O time to system time for the jobs that access the files.

Also, you must consider the volatility of QWIK Disk. If the power goes off or if you perform a cold-start on the system, all disk files, including those in QWIK Disk, are lost.

Table 29–2 highlights the criteria for selecting files for QWIK Disk and the paragraphs following the table describe these criteria.

Table 29–2. QWIK Disk File Section Criteria

File Selection Criteria	
File size	Is it less than QWIK Disk capacity? Certain areas heavily accessed? Can the file overflow to another ID?
File Access	Is it read-only or inquiry? Is it temporary or a work file? Is it an update file? Does it have code file overlays?
File Volatility	Is restart feasible? Will data loss compromise integrity?
File Activity	Does it have a high I/O wait? Are large blocks being transferred? Does it need a high number of I/Os per second?

File Size

Consider the following issues of file size when placing files in QWIK Disk.

File Size

Files that you place in QWIK Disk cannot be larger than the storage available.

File Areas Heavily Accessed

Sometimes you can split a file and put the area used most in QWIK Disk, with the less-used areas on LAK or 5N disk.

File Overflow

When the storage capacity of QWIK Disk fills up, the system seeks space on the default disk subsystem(s). This creates a situation in which there is high-speed access to the part of the file in QWIK Disk and low-speed access to the part of the file that has overflowed to another disk ID.

File Access

Consider the following issues of file access when placing files in QWIK Disk.

Read-Only Files

QWIK Disk offers high speed access to files that are accessed on a read-only basis.

Temporary and Work files

These files generally have a high I/O volume, and you use them as logical extensions of main memory. Therefore, they benefit greatly from the access speeds and transfer rates provided by QWIK Disk. Consider how critical the file is because files in QWIK Disk are more volatile than files on disk or disk pack. DMSII offers an alternative, but recovery is vital.

Update Files

The high-speed access to files that QWIK Disk offers can improve the response time in an online environment where transactions coming from a network are updating files. You must consider the possibility of loss; however, the DMSII recovery capability can provide an option.

Code File Overlay

Use the QWIKPOOL option for program overlays. However, if specific programs have overlays that are too large for QWIKPOOL, consider placing the entire code file in QWIK Disk. For additional information on QWIKPOOL, refer to the discussions of the USE QWIK and LIMIT QWIKPOOL records in Volume 1.

File Volatility

Consider the following issues of file volatility when placing files in QWIK Disk.

Restart

Because QWIK Disk uses RAM memory, information stored in QWIK Disk can be lost when the power fails or when you perform a power down or a cold-start. Could you restore or regenerate files that are lost?

Cold-Start Acceptable

If the MCP resides in QWIK Disk and the system loses power, you must perform a cold-start on the system and reload the MCP to QWIK Disk. Doing so causes the loss of the master disk directory and the loss of all data on disk.

File Activity

Consider the following issues of file activity when placing files in QWIK Disk.

High I/O Wait Time

Files that are accessed randomly and that require a lot of I/Os usually have a high I/O wait time. These files are good candidates for QWIK Disk. If you use multiple buffers, I/O wait time for files that are accessed sequentially can usually be eliminated. If doing so is not possible, you can consider these files for placement in QWIK Disk.

High I/Os Per Second

If the run time of a job is directly proportional to the number of I/Os per second, you can reduce the run time by putting the file in QWIK Disk. Put the smallest files with the highest I/O volume in QWIK Disk. You can also use QWIK Disk in the following cases:

- Files have large blocks to transfer
- Files are small, but frequently accessed
- Files have a high transaction rate that causes high I/O traffic

File Selection Suggestions

The following paragraphs describe the pros and cons of putting various classes of files in QWIK Disk. These paragraphs discuss the following topics:

- Compiler
- SORT
- DMSII
- ISAM
- GEMCOS

For each class of file, an illustration suggests which files to put in QWIK Disk and which to keep on other media.

Compiler

The following paragraphs discuss issues related to compilations.

File Sizing

Table 29–3 shows size estimates per 10,000 COBOL ANSI-74 records as compiled by the COBOL compiler. These figures average about 1 megabyte per 1,000 lines of code, or 500,000 bytes, if you assign the work files to disk. Fifteen megabytes of QWIK Disk can handle approximately 12,000 lines of COBOL code if you send the printer backup files to hard disk. These figures assume that you are compiling only one program at a time.

Table 29–3. Size Estimates for Compiled COBOL Files

File Type	Megabytes per 10,000 Records
Work file	5.6
Source	0.9
Code file	1.2
Listing	1.6
Total	9.3

Placing Compiler Files in QWIK Disk

You control the work file placement for COBOL ANSI-74, RPG and FORTRAN compilers by resetting the compiler option WRKP (RO WRKP). You must also declare the QWIK Disk ID 40/0 as subsystem 8 in the cold-start parameters. This feature of MCP/VS causes all work files to go to disk subsystem number 8 first if there is one. Otherwise, they go to the default disk subsystem.

Because work files are temporary and usually have a high I/O volume, they are good candidates for QWIK Disk.

Put compiler input files and source and COPY libraries in QWIK Disk by using SYSTEM/COPY. Refer to "Operations" later in this section for more information.

QWIK Disk Performance Expectations

Compilation rates vary depending upon listing options, database invocations, and work file locations. Also, the QWIK Disk storage capacity limits the number of compilations that can use QWIK Disk at the same time.

Work files constitute approximately 60 percent of the compiler I/O traffic. If you put work files in QWIK Disk, you should improve compilation rates by approximately 1.3 to 1.4 times.

Some compilers, such as COBOL (900-Series compilers), do few overlays. Placement of the compiler code files in QWIK Disk will produce little improvement on these systems, even if you have not set the QWIKPOOL option.

COBOL copy libraries can cause numerous file openings and closings, in addition to many I/Os. If you use copy libraries heavily, you can improve compilation rates up to 3 times by putting library files in QWIK DISK.

Sort

The following paragraphs discuss issues related to sorting.

File Size

The work files for the SORT intrinsic programs use disk space about 2.25 to 2.6 times the actual size of the input file.

Placing SORT Work Files in QWIK Disk

You can assign SORT work files to disk with the COBOL ANSI-74 SORT SELECT clause or the BPL SORT statement. However, you must declare the QWIK Disk subsystem as the default subsystem to allocate work files to QWIK Disk.

QWIK Disk Performance Expectations

The SORT: program improves marginally when you place its work files in QWIK Disk.

If you place SORT input and output files in QWIK Disk, you will see a significant improvement in performance because of their higher I/O activity.

DMSII

The following paragraphs discuss issues related to DMSII performance.

File Size

DMSII file sizes can vary up to hundreds of megabytes, depending upon file attributes and file population. INDEXED SEQUENTIAL sets and subsets are usually less than 1/4 the size of their associated data sets; therefore, they are good candidates for QWIK Disk.

Placing DMSII Files in QWIK Disk

Declare the location of a DMSII file with the file assignment specification in the DASDL physical attribute description of the structure.

- If you specify the QWIK Disk ID or subsystem number when you first compile the DASDL source, the structure is created in QWIK Disk.
- If you perform a cold-start on the system, you must move the file between the backup medium and QWIK Disk to avoid losing of data.

Alternately, you can place the file on disk with the physical attribute description of the structure. Then move the file to QWIK Disk with the SYSTEM/COPY program and specify the ID or subsystem number of QWIK Disk.

If you have used a DMSII file on another medium, you can change its location to QWIK Disk by changing the physical attribute description in DASDL and recompiling the DASDL source with the $REORGANIZE or $UPDATE option set.

- If you use the $REORGANIZE option, the Reorganize DBP initially moves the file. Then you must archive and restore the file to QWIK Disk.

- If you use the $UPDATE option, you must move the file to QWIK Disk with the SYSTEM/COPY program.

  The DASDL syntax is as follows:

  ```
  FAMILYNAME = DISK { <disk assignment technique> }
  ```

  where <disk assignment technique> can have the following values:

  - `BYID = <integer>`

    The <integer> represents the ID number of QWIK Disk.

  - `BYSUBSYSTEM = DSKn`

    The n represents the subsystem number of QWIK Disk and can have a value from 1 to 8.

QWIK Disk Performance Expectations

A DMSII database can consist of hundreds of structures. You should consider only the smallest structures with the highest I/O volumes for QWIK Disk. Index sequential set and subset files consume much less space than data files. In addition, they are accessed relatively often, so they are good candidates for QWIK Disk.

QWIK Disk performance varies widely, depending on the specific file placement. If you place all or selected database files in QWIK Disk, you can improve performance by a factor of 1 to 4.7 times. The higher the number of database structure buffers, the less the impact of QWIK Disk because of the reduction of physical I/Os. The DMSII DELETE and CREATE-STORE functions usually generate the highest I/O wait time; QWIK Disk has a favorable impact on these functions.

If the information in QWIK Disk is lost, you must use DMSII recovery or database rebuild with the audit trail to recover.

Note: *You should never put the audit trail or the control file in QWIK Disk.*

ISAM

The following paragraphs discuss issues related to the use of ISAM files.

File Sizing

COBOL ANSI-74 and RPG programs offer an indexed sequential file organization (ISAM) and a relative file organization. ISAM clusters consist of a data file and an optional primary key file with up to 98 alternate key files.

ISAM and RELATIVE file sizes vary widely, depending on file attributes and file population. ISAM indexed sequential key files are usually less than 1/4 the size of the data file, and thus are good candidates for QWIK Disk.

Placing ISAM Files in QWIK Disk

You can direct ISAM output data files to QWIK Disk by using the following syntax when you execute the program:

If QWIK Disk is the default subsystem, use this syntax:

```
? FILE <ISAM DATA file-name> = <ISAM DATA file-name> DSK
```

Use this syntax when you execute the program:

```
? FILE <ISAM DATA file-name> = <ISAM DATA file-name> QWK
```

If you enter QWK, the MCP assigns the file to QWIK Disk at open output time.

Refer to Volume 2 for more information about the FILE command.

Note: *The MCP directs the associated key file cluster to the location of the data file.*

You can copy existing ISAM file clusters that are already on disk onto QWIK Disk by using the SYSTEM/COPY program and specifying the QWIK Disk subsystem number as the destination. You can separate key files from large data files and place them in QWIK Disk individually using this method.

If you place ISAM files in QWIK Disk, you must establish a data recovery procedure.

QWIK Disk Performance Expectations

ISAM functions improve if you place data and key files in QWIK Disk. Performance varies, depending on the number of buffers declared; the default value is 3.

The largest improvement from QWIK Disk occurs with the following ISAM functions with high I/O volumes: RANDOM WRITE, DELETE, and RANDOM READ.

GEMCOS

The following paragraphs discuss issues related to GEMCOS files.

File Sizing

The GEMCOS files with the highest activity are the following:

- Control File (GMCTL) Monitor Messages (GMSGS)
- Queue File (GMQ) Format File (GMFMT)
- Audit File (GMAUnn)

The QUEUEFILE and QUEUERECORDSIZE statements in the Transaction Control Language (TCL) source determine the Queue File size. Their size is usually less than 1 megabyte.

The Monitor Message file is read-only and usually contains fewer than 600 80-byte records.

The read-only Format File contains station message formats. The number of TCL formats varies widely if you use the format option.

Placing GEMCOS Files in QWIK Disk

Use the GEMCOS TCL definition language to direct various files to disk in the GLOBAL subsection:

```
CONTROLFILE = {DISK | DSK}
FORMATFILE = {DISK | DSK}
MONMSGFILE = {DISK | DSK}
QUEUEFILE = {DISK | DSK}
```

You can place all GEMCOS files in QWIK Disk with the SYSTEM/COPY program by specifying the QWIK Disk subsystem number or ID number as the destination. You can file equate all GEMCOS files at run time.

QWIK Disk Performance Expectations

Caution

You must never place the audit file in QWIK Disk because MCS recovery depends on it.

You can use the MCS queue file as an overflow for GEMCOS main memory input and station queuing. You should use it infrequently, but you can exploit QWIK Disk fast I/O turnaround in this way to get higher queue throughput during network peak periods.

Monitor Message and Format files are smaller read-only files that you can place in QWIK Disk. If each transaction accesses format files, you can use QWIK Disk as a type of format cache.

The GEMCOS Control file is usually less than a megabyte, but it sometimes creates high I/O counts in an audited environment (SYNChronized recovery, CHECKPOINT recovery, WAITFORAUDIT). If performance bottlenecks become critical, consider moving the Control file to QWIK Disk. You must be aware that loss of the Control file information could compromise network recovery procedures. You might have to use archival recovery to rebuild the control file.

System Configuration Records for QWIK Disk

You must add two special records to your system configuration file to use QWIK Disk.

The DLP special record reserves DLP channel 40 for QWIK Disk. If your system currently has a DLP assigned to channel 40, then you must change the channel number for that DLP.

If you do not change the channel number for that DLP, you will not be able to access it when you install QWIK Disk.

The DISK special record tells the system how much storage QWIK Disk has.

The following system initialization record reserves channel 40 for QWIK Disk:

```
DLP 40 DSK
```

The following system initialization record tells the system how much storage QWIK Disk has and declares QWIK Disk as a nonshared subsystem.

```
DISK 40/0 ID <nn> SUBSYSTEM <nn> 0 <available QWIK Disk sectors (or
segments)>
```

You must always declare QWIK Disk as a nonshared subsystem. For more information about using QWIK Disk on shared systems, refer to "Shared Systems" in this section.

You can enter the TO ALL command to determine the number of available QWIK Disk sectors or segments. If necessary, you can also convert the amount of memory allocated to QWIK Disk into disk sectors by dividing the amount of QWIK Disk memory in digits (not KD or MD) by 200.

Notes:

You must subtract 1 from the number of available QWIK Disk sectors on the disk declaration card because disk addressing is zero relative.

If the DISK card indicates more available sectors than are really available, an I/O error message is displayed when access to that area is attempted.

For more information about system initialization, refer to Volume 1.

Putting the MCP in QWIK Disk

Because QWIK Disk loses all its information in the event of a power failure or power down, you should not put the MCP in QWIK Disk. If the power fails, you would lose the MCP, the disk file directories, and all files on disk. Also, in environments that are memory bound (to the exclusion of 500,000 bytes for QWKMEM), you can expect only marginal improvements over a 206-disk environment.

However, if you do decide to put the MCP in QWIK Disk, you need to add the following records to the system configuration file. These records help you to conserve space in QWIK Disk by directing the system log files to a disk other than to QWIK Disk. In the LOGSUBSYS record, *nn* represents a number other than that of QWIK Disk.

```
CONTROL LOGSUBSYS nn
```

Use the following record to conserve additional space in QWIK Disk by directing the MCP dump file to a medium other than the QWIK Disk subsystem.

```
USE DUMP DISK SUBSYSTEM <nn>

USE DUMP PACK <family>

 or

USE DUMP TAPE
```

Using QWKMEM and QWIKPOOL Options

You can do overlays faster from QWKMEM than from QWIK Disk. When you set the QWKMEM option, you can place code files on a low speed disk and achieve the same performance as you would if they were in QWIK Disk.

For information about the USE QWIK, LIMIT QWIKMEM, and LIMIT QWIKPOOL records and about system initialization, refer to Volume 1.

Shared Systems

In a shared system, you can use QWIK Disk for each system. However, because QWIK Disk uses memory for storage, the files in QWIK Disk can be accessed only by the system that is storing them. QWIK Disk does not have the capability of being exchanged. Therefore, you must declare QWIK Disk as a nonshared ID when you perform a cold-start. For more information about initialization and QWIK Disk, refer to Section 1 of Volume 1.

For consistency, Unisys recommends that all QWIK Disk IDs in a shared environment be declared with the same subsystem number. Doing so enables you to specify QWIK Disk on all systems in the same manner. Also, this shared number enables application programs that specify QWIK Disk to use QWIK Disk on any system on which they run.

Operations

QWIK Disk operation is treated in the same way as any other disk ID. You can load and dump files in the same manner as you would for any other disk.

Maintenance Processor Commands

Use the following maintenance processor commands for QWIK Disk.

```
SET OPTIONS QWIKDISK SET OPTIONS MCP SET OPTIONS LIMIT
```

Use these commands to set system configurations with a form.

```
RESET QWIKDISK
```

Use this command to disable QWIK Disk.

```
SHOW OPTIONS MCP
```

Use this command to display the system configurations.

```
LOAD MCP
```

Use this command to perform a halt/load. The command loads the MCP loader program into the system memory from the MP disk. Later, the MCP loads the MCP code from a default system disk.

```
CLEAR MEM
```

Use this command to clear memory or load memory with a pattern. You should load the control store RAMs (LOAD CS command) before you use this command.

```
MEM ABS
```

Use this command to read from or write to memory at the specified absolute address. The program automatically follows a memory-write command with a memory-read command.

Refer to "Initializing the System" in Volume 1 and to the hardware documentation for your processor for more information about these commands.

Each time you enter any of the first three commands, the system displays the following:

```
QWIKDISK base = <nnnnnnnn>

QWIKDISK sectors = <nnnnnnnn>

QWIKDISK <status>

PHYSICAL MEMORY LIMIT SET TO <nnnnnnnn>
```

Caution on Maintenance Test Commands

The following maintenance test commands can destroy the file that is in QWIK Disk. If you use these commands, the system displays this warning:

> **Caution**
>
> This command will destroy memory/QWIK Disk contents.

Before the system can execute the command, you must give it permission by appending the word CLOBBER. If you use the word CLOBBER the first time you use the command, the system does not display the warning and executes the command.

You should copy the files in QWIK Disk to a safe medium before you use any of the following commands:

- CHAINTO - T7 CLOBBER
- TEST CH TO -T7 CLOBBER
- START MODTST <SPATH> CLOBBER
- CLEAR 0 CLOBBER
- CLOCK 0 CLOBBER
- TEST MEM CLOBBER
- DISPLAY TO - T7 CLOBBER
- MSEL, MSHIFT, MCKEN, MGCKEN

Powering On the System

Each time you power on the system, you must clear all system memory. You do so with the CLEAR MEM command. You must do this regardless of whether you plan to cold-start the system. For more information, refer to Volume 1.

Halt/Load

When you halt/load a system that has QWIK Disk, the QWIK Disk memory area is rewritten and cleared of any transient single bit memory errors. No information in QWIK Disk is lost or changed, but this operation can make the halt/load take longer to complete.

Putting Files in QWIK Disk

Perform the following steps to place files in QWIK Disk:

1. Move selected files to QWIK Disk before you start a program. You can move them with the SYSTEM/COPY program or create them programmatically in QWIK Disk.
2. Execute the program. It will receive performance benefits from accessing files from QWIK Disk.
3. When the program finishes, move files (other than read-only files) from QWIK Disk to other media to archive or to update backup copies.

You can move files with the SYSTEM/COPY program or with a Work Flow Language (WFL) program that moves the files to QWIK Disk, starts the job, and moves the files back to the original medium when the job finishes. You do not need to move temporary work files to or from other media.

Using SYSTEM/COPY

Use the SYSTEM/COPY program to copy files to QWIK Disk. Specify the QWIK Disk ID number or the subsystem number as the destination. Refer to Section 24, "SYSTEM/COPY–File Transfer Utility Program," for a description of this program. Also, refer to the COPY command in Volume 2 for the command syntax.

Examples

```
COPY MYFILE FROM DISK TO DISK (ID 1)
```

where QWIK Disk is ID 1

```
COPY MYFILE FROM DISK TO DISK (SUBSYSTEM 8)
```

where QWIK Disk is disk subsystem 8

Loading and Unloading Files from QWIK Disk

Loading 15 megabytes of QWIK Disk from PE tape can take from two to two-and-one-half minutes.

Firmware

The firmware you load from the floppy PANDMV redirects I/Os to a "soft" channel/unit 40/0. Externally, this configuration looks like a 5N disk.

Programming Considerations

The following paragraphs describe the programming considerations for QWIK Disk.

Programmatic File Creation in QWIK Disk

You can create files programmatically in QWIK Disk with the file attributes or file declarations available with individual programming languages.

QWIK Disk as Default Disk

Unisys does not recommend that you use QWIK Disk as your default disk, because QWIK Disk has a limited capacity. It can fill up if all the files created on disk, without a specific subsystem or ID number, go to QWIK Disk. If QWIK Disk does fill up, the system looks for available space on nondefault subsystems and displays the following message:

```
NEED nnnn DSK SEGS FOR <file-id> <prog-id> = <mix number>
```

Work File Subsystem Default

A feature of the MCP directs all compiler work files to the disk subsystem number 8 if there is one. If not, the system places the work files as it did for prior releases of the MCP.

You must set the MCP option WRKP to take advantage of this option.

This feature enables you to automatically direct work files to QWIK Disk by declaring it as subsystem number 8. Because work files are temporary and usually have a high I/O volume, they are good candidates for QWIK Disk.

When the system displays the following message indicating that QWIK Disk is full, you can direct the overflow to another medium.

```
** NO USER DISK <job-specifier>
```

File Equation to QWIK Disk

QWK is the hardware type designation for QWIK Disk. Use the following syntax to perform file equation operations for QWIK Disk:

```
FILE <file-name> = <file-name> QWK
```

This syntax causes the MCP to assign the file to QWIK Disk at open (output) time. If there is no QWIK Disk, the file goes to the default disk. WFL supports this syntax.

Random and Sequential I/O

Random and sequential I/O access times are the same in QWIK Disk. The use of random or sequential I/O causes data to be stored in RAM memory. Therefore, the data transfer time is not dependent on seek time or latency positioning.

The amount of time it takes to complete a QWIK Disk I/O equals the amount of time it takes to transfer data. You can approximate the QWIK Disk I/O time by dividing the block length in bytes by the QWIK Disk data transfer rate of 1.5 megabytes per second.

Figure 29–9 shows the BLOCKSIZE and time factors for completing I/Os to QWIK Disk.

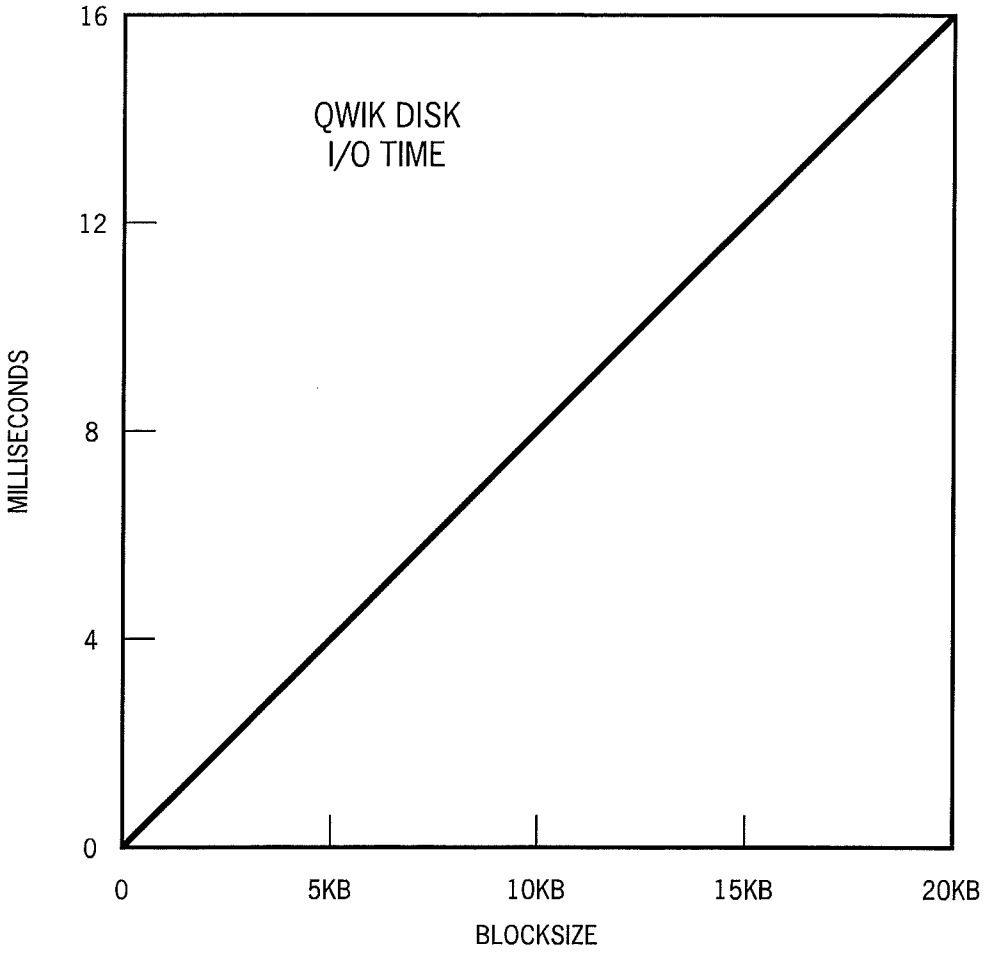


Figure 29–9. BLOCKSIZE and Time Factors for Completing I/Os to QWIK Disk

Code File Overlays

The QWIKPOOL option provides the best performance for code file overlays. However, you can use QWIK Disk in memory constrained environments or in situations where many overlays exceed the size of the QWIKPOOL PAGE. If you direct the oversize overlays to QWIK Disk, you can optimize the I/O wait time that would have otherwise been spent accessing the overlay on conventional disk. A QWIK Disk overlay of 9000 bytes costs about 6 milliseconds.

Deviation from Standard Disk Operation

There is one other deviation from standard disk operation. Normally, when a write operation to disk does not fill out a sector, the point from the end of the data through to the end of the sector is filled with zeros. QWIK Disk does not do this. For QWIK Disk, the write operation ends when the data runs out. The bytes remaining in that block remain set as they were set before the write operation.

Error Conditions

The following paragraphs describe QWIK Disk error conditions.

Memory Error

The following error message indicates that an I/O operation tried to exceed the end of memory. The system thinks the memory limit is higher than it really is. This happens at cold-start time when the DSK card indicates that there were more segments available in QWIK Disk than there really are.

```
Not ready (C000 8000)
```

Result Descriptors

The following QWIK Disk hardware exceptions can appear. If the MCP encounters any of these errors, they appear in the maintenance log (MLOG).

The following error message indicates that a read operation encountered a double bit error in QWIK Disk memory.

```
Data read error (C080 0021 0002 0000)
```

The following error message indicates that a write operation encountered a double bit error in MCP memory.

```
Memory error result (C400 0000 0000 0000)
```

Single Bit Errors

All single bit memory errors encountered in QWIK Disk are recorded in the maintenance log (MLOG) in the same way as errors in MCP memory areas are recorded.

Caution

In the event of a power failure or if the system is turned off, QWIK Disk loses all the information it is storing. Because the V Series QWIK Disk does not provide battery backup, you might want to use an Uninterruptible Power System (UPS) as backup. In the event of a power failure, you could use the power from the UPS to copy the QWIK Disk files to tape, disk pack, or hard disk.

Section 30
SHARED—Shared Systems and Devices

Overview

Shared systems (SHARED) provide a method for multiple programs (including the MCP) to access disk and disk pack files concurrently. In addition, up to four systems can concurrently access files resident on shared disk or disk pack media.

Specifically, a shared system accomplishes some or all of the following goals:

- Permits multiple programs to access a common file concurrently. The SHARED feature provides additional I/O operations and a block lockout table (BLT) to lock files at block level, which prevents simultaneous updating. The following example illustrates block-level locking:

  Program A reads a record of a file.

  Program B reads the same record.

  Program A modifies some fields and writes the record.

  Program B modifies some fields and writes the same record, thus destroying the update accomplished by Program A. To prevent this problem, each program uses shared I/O operations to lock out other programs while it updates the record.

  Here is another example:

  Program A reads a record, using a read with lock operation.

  Program B tries to read the same record in the same file, but cannot because program A has already locked the record. Program B must wait until program A unlocks the record.

  Program A modifies some fields, writes the record, and then unlocks it.

  Program B can now lock and read the record, update some fields, and then write the record and unlock it.

- Permits multiple systems to share disk and disk pack resources. Hardware that is called a shared systems processor (SSP) accomplishes this sharing and acts as a common repository for all locks and contentions between the shared systems.

- Provides a growth path for a memory-bound environment. Hardware memory limitations are often overcome by adding another mainframe tied to the existing peripherals, which provides added memory resources to the user environment.
- Provides a growth path for a system bound environment. The addition of another mainframe tied into the existing peripheral network augments the processing power of the user environment.
- Provides redundancy. Many users require a fault-tolerant environment. Shared systems permit the duplication of mainframe resources (system, memory, DLPs, and so forth) either in a passive or active state. Often one system is dedicated to online activity while the other is processing batch mode. If the critical online system should fail, the batch system can take over that responsibility with minimal disruption.

A special set of file I/O operations regulates concurrent file access. By using shared file I/O operations, programs can read, seek, and write file blocks in a manner that prevents data corruption by other programs that are also using shared file I/O operations. If a program opens a file and accesses it without using shared file I/O operations, the file is not protected and the data can be corrupted.

The following rules apply to shared file I/O operations:

- If one program opens a file for shared access, all programs should open the file shared. You should follow locking and unlocking protocols to prevent simultaneous updates.
- All programs must declare the same record size and blocking factor for the file they are sharing.
- Shared files must be declared random access.
- A program can relock a record (for example, LOCK followed by READ LOCK).
- A program is terminated if it attempts to unlock a block that it does not have locked.
- A program is terminated if it attempts to write to a block that is locked by another program. You should do some locking operations (for example, LOCK, READ LOCK, and so forth) before the write.

A program performs lock/unlock operations on a record level, but because the MCP deals with blocks of records for I/O, the locking operation is accomplished on the block level. This is why all accessing programs must refer to a shared file with the same record size and blocking factor.

Figure 30–1 illustrates concurrent file access.

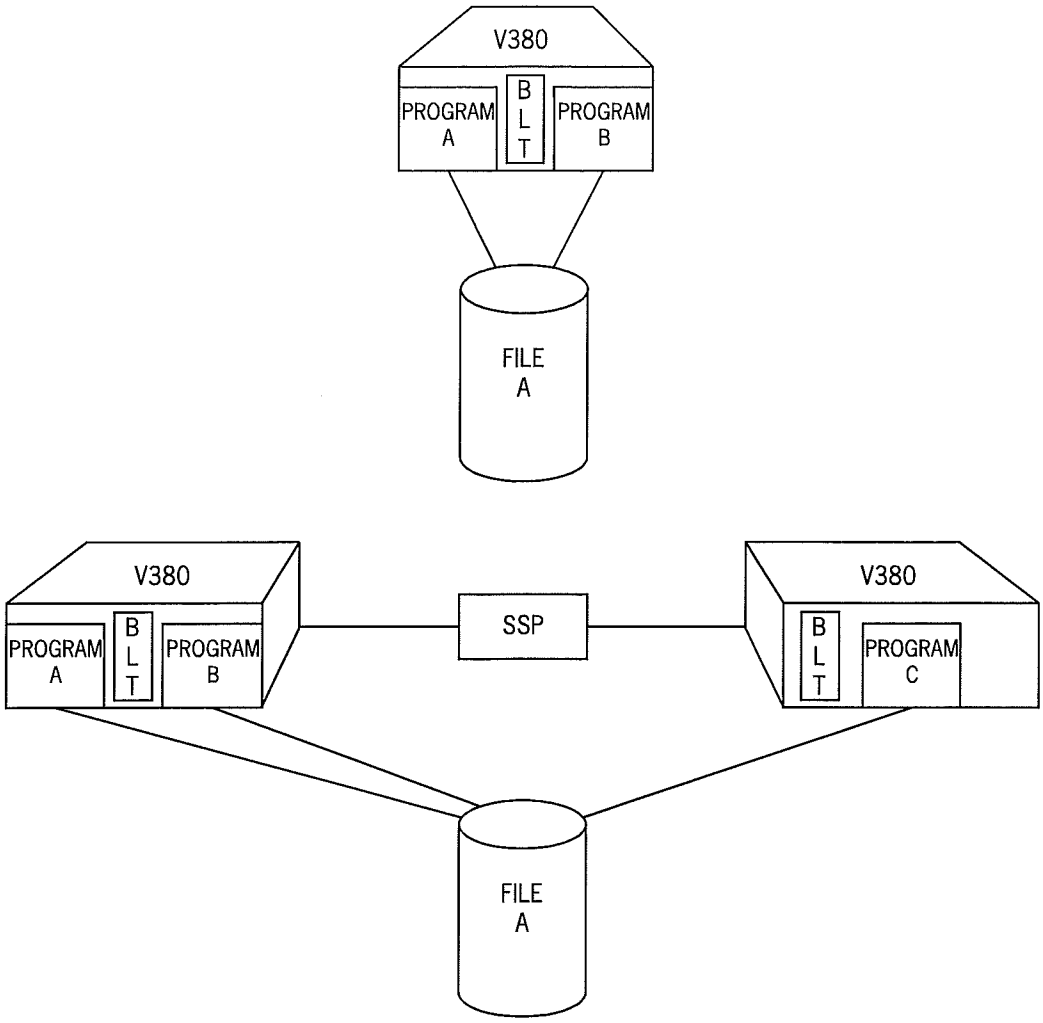


Figure 30–1. Concurrent File Access

Only the COBOL and BPL programming languages support shared file I/O operations. The types of shared file I/O operations and the corresponding syntax appear in alphabetical order in Table 30–1.

Table 30–1. I/O Operations

Function	BPL	COBOL	BCT/Variant
LOCK	LOCK	READLOCKONLY	0114/4
LOCK NO CONTEND	LOCK SEEK	SEEKLOCKONLY	0314/4
UNLOCK	UNLOCK	UNLOCK	0234/8
READ	READ	READ	0114/0

continued

Table 30-1. I/O Operations (cont.)

Function	BPL	COBOL	BCT/Variant
LOCK, READ	READ LOCK	READLOCK	0114/8
LOCK, READ, UNLOCK	READ NO LOCK	READLOCKWAIT	0114/C
SEEK	SEEK	SEEK	0314/0
LOCK SEEK	SEEK LOCK	SEEK LOCK	0314/8
LOCK, SEEK, UNLOCK	SEEK NO UNLOCK	SEEKLOCKWAIT	0314/C
WRITE	WRITE LOCK	WRITELOCK	0234/4
WRITE, UNLOCK	WRITE	WRITE	0234/0

Single-System Shared Environment

A single-shared system does not need hardware modification to perform concurrent file access. All disk and disk pack media are capable of storing shared files, and the system runs shared by default. An SSP is not required in a single-system shared configuration.

All programs sharing a file must declare the same blocking factor for the file. When you access the file, you must declare it random access and shared.

Multisystem Shared System Environment

In a multisystem shared environment, you declare and name an SSP with the DLP configuration record. All disks and disk packs that the multiple systems are to access should be set up as SHARED VIA <SSP name>. Not all media must be shared. The following paragraphs describe shared configuration rules:

- If any disk is shared, subsystem 1 must be shared because the MCP and disk directory must be available to all systems.
- Additional disks can be shared, but all disks belonging to a specific subsystem must have the same shared attributes.
- QWIKDISK cannot be shared, because it is not accessible by other systems.
- All disk packs belonging to the same family must have the same shared attributes, either shared or non-shared.

Because the MCP considers all disk and disk pack media eligible to store shared files by default, file placement can no longer be guaranteed. Thus, the SHARED open attribute merely enables the program to issue shared file I/O operations to that file. This feature is different from that of previous MCPs, which only allowed shared file assignment to media declared SHARED.

All programs accessing a shared file must do so consistently. That is, erroneously the same file could be opened simultaneously shared and non-shared, thus leading to the loss of the record protection that sharing is intended to afford. This loss is consistent with previous MCPs.

Types of Shared Systems

"Shared" refers to the ability of programs to access the same disk or disk pack file simultaneously, yet be afforded protection from simultaneous updating because of a shared record locking/unlocking protocol.

The following paragraphs describe the types of shared system configurations that are available.

Single-System Shared

This term refers to a shared system that uses only one mainframe. Multiple programs are able to access a common file simultaneously. Under previous MCPs, single-system shared configuration was invoked by the SHARED option of the DSKn and PACK MCP configuration records and by the SHRD system option. An MCP/VS 2.0 or later version eliminates these requirements. The system is considered to be single-system shared by default.

Multisystem Shared

This term refers to a shared system that is extended to multiple mainframes.

A multisystem shared configuration requires the addition of a shared systems processor (SSP) to handle the locking/unlocking between systems. Up to four V Series mainframes can be configured in a multisystem shared environment.

Shared Disk Only System

This term refers to a multisystem shared configuration that has one or more disks as common resources and no shared disk pack resources. The MCP disk must be shared. Each system runs the same MCP code file and shares the disk directory and available table. In this configuration, subsystem 1 must be shared and additional disks can be shared on a subsystem level.

Shared Disk Pack Only System

This term refers to a multiple-system configuration that has one or more disk packs as common resources and no shared disk resources. The peculiarity here is that each system runs a separate copy of the MCP. The copies of the MCP must be the same version and release level.

Figure 30–2 illustrates a shared disk pack only system.

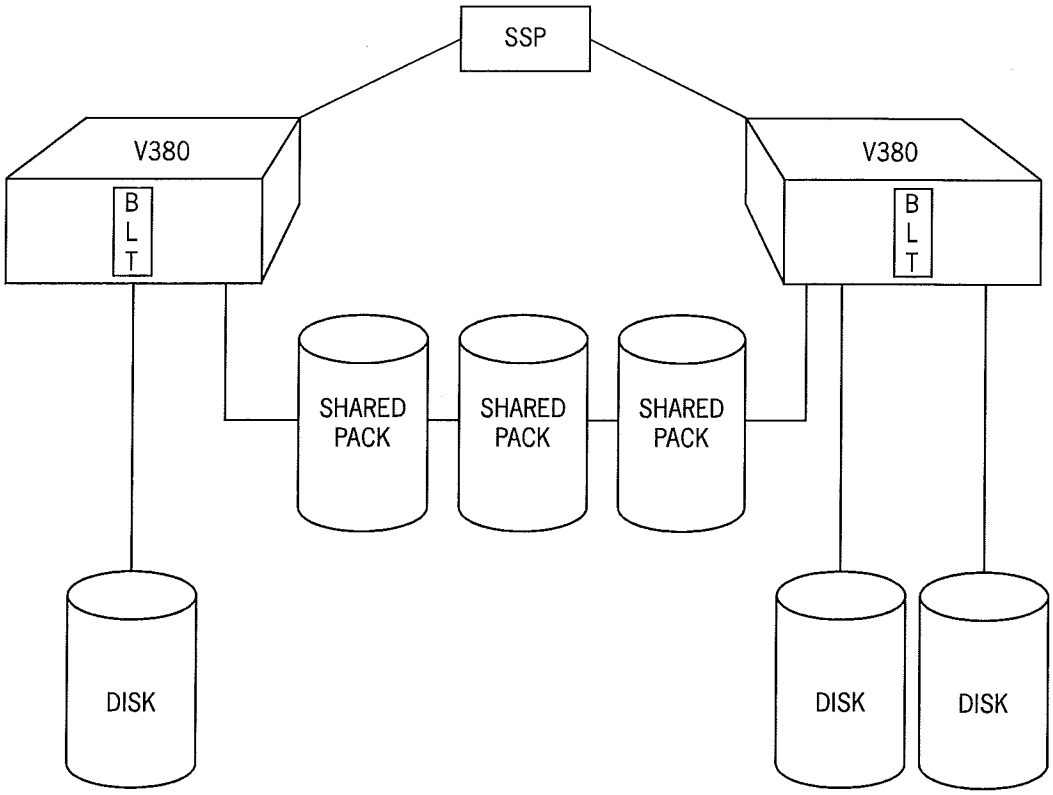


Figure 30–2. Shared Disk Pack Only System

Shared Disk and Disk Pack System

This system is a multisystem shared configuration that has one or more disks and one or more disk packs as common resources. The MCP disk must be shared. Each system runs the same MCP code file and shares the disk directory and available table. Additional disks can be shared at the subsystem level, and disk packs can be shared at the family level.

Figure 30–3 illustrates a shared disk and disk pack system.

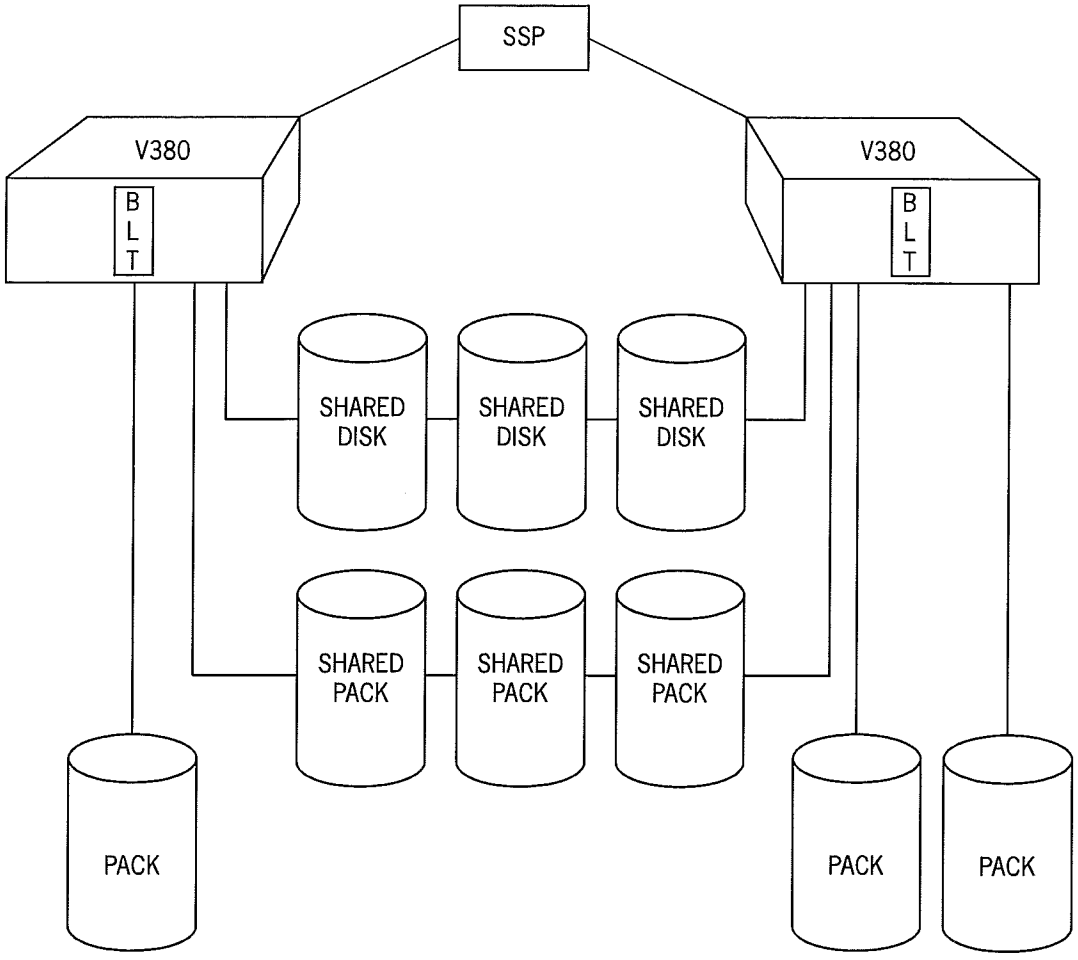


Figure 30–3. Shared Disk and Disk Pack System

Definition of Shared File I/O Operations

The following paragraphs define the shared file I/O operations that appear in Table 30-1.

BLT Contention

BLT contention occurs when one program wants to access a file block that another program running on the same system has locked. Refer to "Overview" earlier in this section for more information about the block lockout table (BLT). When the programs are running on different systems, this situation is called SSP contention.

Contention

When one or more programs attempt to lock a block that is locked by another program, the programs are suspended. They must wait until the block is unlocked. A suspended program is said to be contending for the block.

The amount of time a program can contend for a file block is called the contention cycle. If the block does not become available during the contention cycle, the operating system intervenes and either reinstates the program at a stalemate-use routine, if one has been coded into the program, or terminates the program.

The default contention cycle is 10 seconds. Use the LIMIT DELAY record in the system configuration file to override the default. Refer to Volume 1 for more information.

If two or more programs are waiting in contention, the program with the highest system priority gets the block first. Use the PRIORITY command to set the priority of a program. The system then restarts the contention cycle for the other programs that are contending for the block.

Contention Cycle

The contention cycle is the period of time that a program can contend for a file block. After this period of time expires, the system either reinstates the program at a stalemate-use routine, if one was coded into the program, or it terminates the program. Refer to "Contention" earlier in this section for more information.

LOCK

This shared file I/O operation locks a block and prevents other programs that are using shared file I/O operations from locking the block. The block remains locked until it is unlocked. If the block is locked already by another program, a program must contend for the block.

LOCK NO CONTEND

This shared file I/O operation performs the function of the normal lock operation, but does not contend for the lock if it is locked by another program. In this case, the MCP immediately reinstates the program at a stalemate use routine, if one was coded into the program, or terminates the program.

LOCK, READ

This shared file I/O operation locks a block and then reads it. The block remains locked until it is unlocked.

LOCK, READ, UNLOCK

This shared file I/O operation locks a block, reads it, and then unlocks it.

LOCK SEEK

This shared file I/O operation locks a block and reads it to a buffer for later use. The block remains locked until it is unlocked.

LOCK, SEEK, UNLOCK

This shared file I/O operation locks a block, reads it to a buffer for later use, and then unlocks it.

MAILBOX

This term refers to an intersystem message-passing mechanism used to synchronize systems for MCP functions that require exclusive access to shared resources. For MCP/VS 2.0 or later, the mailbox was moved into the SSP and is no longer on disk. The record in the system configuration file that declared a MAILBOX for previous MCPs is not used with MCP/VS 2.0 or later. However, the DLP SSP record in the system configuration file must include the MAILBOX attribute for the SSP that you are using and that contains the mailbox.

READ

This I/O operation reads a file block even if another program has locked that block.

RLT

RLT is the record lockout table, which is the MCPIX/MCP/VS Mark 1.0 version of the MCP/VS Mark 2.0 or later block lockout table (BLT).

Stalemate

This situation occurs when two or more programs cannot continue processing because they are trying to lock mutually exclusive blocks within a file. Because they are waiting for each other, their contention cycle expires and the system either reinstates them at a stalemate-use routine, if one has been coded into the program, or it terminates the programs.

Here is an example of a stalemate:

Program A locks record 10 of a file.

Program B locks record 20 of the same file.

Program A attempts to lock record 20 of the same file, fails, and goes into a contention condition.

Program B attempts to lock record 10 of the same file, fails, and goes into a contention condition.

The period of time these programs can wait in contention (10 seconds or the amount specified by the LIMIT DELAY record) expires. The MCP intervenes and either terminates the programs or reinstates them at a stalemate use routine.

The MCP does not detect true stalemate conditions. It merely sets a limit to the time that a program can wait for a lock without being given an opportunity to retry it because another program has unlocked. The program that owns the lock may not have sufficient system resources to perform an unlock, so the program does not get

the opportunity to retry. Refer to "Stalemate Use Routine" in this section for more information.

SSP Contention

This condition occurs when a program running on one system wants to lock a block that a program running on another system has locked. If the programs are running on the same system, it is called BLT contention. Refer to "Contention Cycle" and "Shared System Processor" in this section for more information.

Stalemate Use Routine

This mechanism enables you to reinstate a program at a routine instead of terminating it when the program is in contention for a period of time exceeding the contention overtime cycle. The sophistication of the stalemate-use routine can vary from simply ignoring the condition and returning to the main program logic, to backing out a previous series of updates and allowing the other programs to complete their operation.

SEEK

This operation locates a block and reads it to a buffer for later use even if another program has locked that block.

UNLOCK

This shared file I/O operation unlocks a locked block. If the program that uses this operation does not already have the block locked, the program is terminated.

WRITE

This I/O operation expects the relevant block to be locked already. However, it attempts the lock if it is unlocked. If this attempt fails, the program is terminated. If the block was already locked by the program or if the attempted lock was successful, the block is written and then unlocked.

WRITE NO UNLOCK

This I/O operation requires that the block be locked already by the program. Otherwise, the program is terminated. Also, when the block is written, it is left locked. These WRITE operations and requirements are consistent with previous MCPs.

Components of a Shared System

The following paragraphs describe the components of a shared system.

BLOCK LOCKOUT TABLE (BLT)

This component keeps track of the file blocks that are locked by which programs. The block lockout table (BLT) resides in memory, and the MCP updates it every time a program locks or unlocks a file block. The MCP provides a BLT with a sufficient number of entries to handle most circumstances. However, this default setting can be changed with the MCP configuration record LIMIT BLT. Refer to the LIMIT BLT record in Volume 1 for more information.

DISK

From the single-system perspective, all disk is considered eligible for shared file access. From a multisystem perspective, disk sharing is determined at the subsystem level. This means that all members of the same disk subsystem must have the same shared attributes. The SHARED option on the MCP configuration record DISK declares a disk to be multisystem shared. Refer to Volume 1 for more information.

DISKPACK

From the single-system perspective, all disk pack is considered eligible for shared file access. From a multisystem perspective, disk pack sharing is determined at the family level. You can declare some disk pack units shared, and you can declare other disk pack units not shared. All members of the same disk pack family should be either shared or not shared.

The SHARED option of the record called PACK in the system configuration file declares a disk pack to be multisystem shared. When you declare a disk pack shared, you must assign the same logical ID number and the same SSP name for each system.

FAMILY NAME

Individual disk packs can be organized into families (groups). A family name is the name that you assign to a group of disk packs that is organized into a family. The MCP can then direct individual files to and store individual files on these groups of disk packs according to the family name.

LOGICAL SUBSYSTEM

This component represents a family of disk devices. Disk pack devices can be grouped into families for file assignment; similarly, disks can be grouped into subsystems. Subsystems have attributes such as default (available for generic file assignments) and shared (accessible to multiple systems). Use the SUBSYSTEM option of the record DISK in the system configuration file to define a disk subsystem.

MASTER CLEAR

This component is an electronic signal that the system uses to initialize a DLP and its associated hardware. Master Clear does not clear the SSP. When you do a BASE CLEAR, make sure that none of the systems attached to the base are using the SSP because it will cease to function and have to be reloaded.

When you enter TERM on the V Series, you generate a MASTER CLEAR. You generate BASE CLEAR with the switch on the base.

PHYSICAL SUBSYSTEM

This component is a group of disks and disk packs, or both, connected to the system with the same DLP.

SHARED SYSTEM PROCESSOR (SSP)

This component is a special hardware device that controls concurrent file access for programs running on two to four V Series systems. It is a specialized four-card DLP that resides in a dedicated base shared by the systems. The shared system processor (SSP) keeps track of the file locks held by a system and the systems that might be contending for those locks. It also stores the intersystem mailbox.

You must load the SSP with the proper firmware before you can use it. MCP/VS Mark 2.0 or later requires firmware level SSP302 or greater. You must never reload the SSP while it is in use by other systems because all locks and contentions would be purged. Refer to Section 12, "LOADFW—Offline Firmware Loader Program," for more information about how to load firmware to SSPs.

Refer to Volume 1 for more information about the system configuration file record called SSP DLP.

Configuration and Initialization of Shared Systems

The following paragraphs describe configuring and initializing single and multisystem shared systems.

Single System Shared

Initialization of a single-system shared system is similar to a normal system initialization because the system is designed to support this feature by default. Note that this differs from previous MCPs, where disk and disk packs used to store shared files were declared shared. The only modifications might be to increase or decrease the number of BLT entries or the contention overtime cycle with the LIMIT BLT and LIMIT DELAY configuration records. The defaults for these records will suffice for most operations.

Multiple-Shared System

There are several additions and modifications required for the initialization of multiple shared systems. You might need to add or modify the following records in the system configuration file for shared operations.

- **SYSTEM record.** This record identifies a system configuration file in stacked configuration decks.
- **CONNECT record.** This record identifies the systems to be connected to this system to comprise a multisystem shared system.
- **DISK record.** This record declares a disk unit as a multisystem shared through a particular SSP.
- **PACK record.** This record declares a disk pack unit as a multisystem shared through a particular SSP.
- **UNIT SHARED record.** This record declares a tape unit as a multisystem shared.
- **SSP DLP record.** This record declares the SSP to use for multisystem shared configurations.
- **LIMIT BLT record.** This record declares the number of entries in the Block Lockout Table.
- **LIMIT DELAY record.** This record sets the contention cycle.

These records enable the MCP and other programs to access files concurrently and to share peripherals. The system uses these records when you cold-start the system. Refer to Volume 1 for more information.

You can maintain shared configuration files in one of the following ways:

- Each system can have its own file. Such is the case when the CONFIG utility is used to maintain the file on floppy disk or when the systems are only sharing disk packs.
- One system can input and build the configuration files for all shared systems. The system does so with a stacked configuration deck, where the configuration decks follow one another and are identified by the SYSTEM record.

The following examples illustrate a two-system shared disk and disk pack configuration. In example file 1, the system is named NSHARED and has the system number 0. In example file 2, the system is named MSHARED and has the system number 2. Finally, the example of a stacked deck covers both file 1 and 2. It would be input when system NSHARED is initialized. It generates the system configuration files (on disk) for NSHARED (where the file is named CONFG0) and MSHARED (where the file is named CONFG2).

File 1

In this example, the system configuration file is set up for a system named NSHARED, with the hardwired system number 0.

```
1 ...HOSTNAME NSHARED
2 ...CONNECT 2
3 ...DLP 34 SSP SSPAAA SSP302 MAILBOX
4 ...DLP 04 DSK HSTLQG
5 ...DISK 4/0 ID 01 SUBSYSTEM 1 DEFAULT SHARED VIA SSPAAA
6 ...DISK 4/1 ID 02 SUBSYSTEM 1 DEFAULT SHARED VIA SSPAAA
7 ...PACK 4/2 ID 03 SHARED VIA SSPAAA
8 ...PACK 4/3 ID 04 SHARED VIA SSPAAA
9 ...DLP 06 MPE
10 ...UNIT 6/1 MPE SHARED.
11 ...UNIT 6/2 MPE SHARED.
12 ...UNIT 6/3 MPE SHARED.
13 ...UNIT 6/4 MPE SHARED.
14 ...DLP 26 MPE
15 ...UNIT 26/0 MPE
16 ...DLP 13 PRN PRN256
17 ...UNIT 13/0 PRN256 PRN
18 ...DLP 01 CRD
19 ...UNIT 1/0 CRD
20 ...DLP 10 CONSOLE
21 ...UNIT 10/4 SPOB ODT LEVEL 9 AD SK 1 HDR MSG D 4
22 ...DLP 12 UNILINE.
23 ...UNIT 12/0 SPOC OCS LEVEL 9 AD SK 1 HDR (WM AM) (MTP DSK DPK PRN)
24 ...USE SLOG AUTO 10000 WRAP
25 ...USE RLOG
26 ...USE BOJ
27 ...USE EOJ
28 ...USE PBD
29 ...USE DUMP DISK
30 ...LIMIT DELAY 30
31 ...STOP
```

File 2

In this example, the system configuration file is for a system named MSHARED, with the hardwired system number 2.

```
33 ...HOSTNAME MSHARED
34 ...CONNECT 0
35 ...DLP 34 SSP SSPAAA SSP302 MAILBOX
36 ...DLP 04 DSK HSTLQG
37 ...DISK 4/0 ID 01 SUBSYSTEM 1 DEFAULT SHARED VIA SSPAAA
38 ...DISK 4/1 ID 02 SUBSYSTEM 1 DEFAULT SHARED VIA SSPAAA
39 ...PACK 4/2 ID 03 SHARED VIA SSPAAA
40 ...PACK 4/3 ID 04 SHARED VIA SSPAAA
41 ...DLP 06 MPE
42 ...UNIT 6/1 MPE SHARED.
43 ...UNIT 6/2 MPE SHARED.
44 ...UNIT 6/3 MPE SHARED.
45 ...UNIT 6/4 MPE SHARED.
46 ...DLP 01 CRD
47 ...UNIT 1/0 CRD
48 ...DLP 13 PRN PRN256
49 ...UNIT 13/0 PRN256 PRN
50 ...DLP 10 CONSOLE
51 ...UNIT 10/4 SPOB ODT LEVEL 9 AD SK 1 HDR MSG D 5
52 ...DLP 12 UNILINE.
53 ...UNIT 12/0 SPOC OCS LEVEL 9 AD SK 1 HDR (WM AM) (MTP DSK DPK PRN)
54 ...USE SLOG AUTO 10000 WRAP
55 ...USE RLOG
56 ...USE BOJ
57 ...USE EOJ
58 ...USE PBD
59 ...USE DUMP DISK
60 ...LIMIT DELAY 30
61 ...STOP
```

Stacked Deck

In this example, the system configuration file would be input to system N when it is cold-started. That system would then build the configuration files for both systems, NSHARED and MSHARED.

```
1 ...HOSTNAME NSHARED
2 ...CONNECT 2
3 ...DLP 34 SSP SSPAAA SSP302 MAILBOX
4 ...DLP 04 DSK HSTLQG
5 ...DISK 4/0 ID 01 SUBSYSTEM 1 DEFAULT SHARED VIA SSPAAA
6 ...DISK 4/1 ID 02 SUBSYSTEM 1 DEFAULT SHARED VIA SSPAAA
7 ...PACK 4/2 ID 03 SHARED VIA SSPAAA
8 ...PACK 4/3 ID 04 SHARED VIA SSPAAA
9 ...DLP 06 MPE
10 ...UNIT 6/1 MPE SHARED.
```

```
11 ...UNIT 6/2 MPE SHARED.
12 ...UNIT 6/3 MPE SHARED.
13 ...UNIT 6/4 MPE SHARED.
14 ...DLP 26 MPE
15 ...UNIT 26/0 MPE
16 ...DLP 13 PRN PRN256
17 ...UNIT 13/0 PRN256 PRN
18 ...DLP 01 CRD
19 ...UNIT 1/0 CRD
20 ...DLP 10 CONSOLE
21 ...UNIT 10/4 SPOB ODT LEVEL 9 AD SK 1 HDR MSG D 4
22 ...DLP 12 UNILINE.
23 ...UNIT 12/0 SPOC OCS LEVEL 9 AD SK 1 HDR (WM AM) (MTP DSK DPK PRN)
24 ...USE SLOG AUTO 10000 WRAP
25 ...USE RLOG
26 ...USE BOJ
27 ...USE EOJ
28 ...USE PBD
29 ...USE DUMP DISK
30 ...LIMIT DELAY 30
31 ...SYSTEM 2
32 ...HOSTNAME MSHARED
33 ...CONNECT 0
34 ...DLP 34 SSP SSPAAA SSP302 MAILBOX
35 ...DLP 04 DSK HSTLQG
36 ...DISK 4/0 ID 01 SUBSYSTEM 1 DEFAULT SHARED VIA SSPAAA
37 ...DISK 4/1 ID 02 SUBSYSTEM 1 DEFAULT SHARED VIA SSPAAA
38 ...PACK 4/2 ID 03 SHARED VIA SSPAAA
39 ...PACK 4/3 ID 04 SHARED VIA SSPAAA
40 ...DLP 06 MPE
41 ...UNIT 6/1 MPE SHARED.
42 ...UNIT 6/2 MPE SHARED.
43 ...UNIT 6/3 MPE SHARED.
44 ...UNIT 6/4 MPE SHARED.
45 ...DLP 01 CRD
46 ...UNIT 1/0 CRD
47 ...DLP 13 PRN PRN256
48 ...UNIT 13/0 PRN256 PRN
49 ...DLP 10 CONSOLE
50 ...UNIT 10/4 SPOB ODT LEVEL 9 AD SK 1 HDR MSG D 5
51 ...DLP 12 UNILINE.
52 ...UNIT 12/0 SPOC OCS LEVEL 9 AD SK 1 HDR (WM AM) (MTP DSK DPK PRN)
53 ...USE SLOG AUTO 10000 WRAP
54 ...USE RLOG
55 ...USE BOJ
56 ...USE EOJ
57 ...USE PBD
58 ...USE DUMP DISK
59 ...LIMIT DELAY 30 60 ...STOP
```

Section 31
Pack Subsystems

This section describes the concepts, structure, and operations of pack subsystems for V Series systems.

Disks and Packs

Disks and packs are similar types of storage media. The difference between the two is the number of bytes per sector:

- Disks are formatted into 100 bytes per sector.
- Packs are formatted into 180 bytes per sector.

Note that the terms *pack* and *disk pack* are interchangeable.

In many cases, disks can be formatted as packs and packs can be formatted as disks. Disks and packs are formatted with the Initialization, Verification, and Relocation (IVR) functions.

V Series systems supports one type of disk that can only have 100-byte sectors: QUIK Disk.

Other disk types are *soft-sectored*; they can be formatted as 180-bytes-per-sector media. A pack that is formatted into 100 bytes per sector is known as a *look-alike*, or *LAK*, device. You can use either the DISPKV utility program or the PTD program to perform IVR functions on a pack.

Table 31–1 lists for each type of pack device which program is used (DISPKV or PTD) to perform IVR functions, indicates whether the pack can be formatted as an LAK device, and specifies whether or not the pack is removable from the pack drive.

Table 31–1. Capabilities and IVR Functions for Pack Types

Pack Type	LAK Capable?	Removable?	IVR Functions
206	Yes	Yes	DISPKV
207	Yes	No	DISPKV
225	Yes	Yes	DISPKV
235	Yes	Yes	DISPKV
659	No	No	DISPKV
677	Yes	Yes	DISPKV
3680	No	No	PTD
3682	No	No	PTD
MD4	Yes	No	PTD
MD8	Yes	No	PTD
SCSI	Yes	No	PTD

Unisys V Series systems treat various models of packs, regardless of different capacities and performance capabilities, as logically identical.

Look-alike (LAK) packs are not discussed further in this section.

Development of V Series Disk and Pack Subsystems

This section discusses the evolution of disk and pack subsystems for Unisys B 2000/B 3000/B 400 and V Series mainframes. This information may be helpful in understanding the current development of these subsystems.

The original Medium System mainframes used only 100-byte disks as primary storage media. Each disk was represented by an *Electronics Unit (EU)* and one or more *Storage Units (SUs)*. The EU was the smallest unit to which files could be assigned. This term is still used today to denote an individual disk unit.

Multiple EUs could be grouped together with an exchange cabinet to form a physical subsystem. With MCP release 6.6, this subsystem concept was changed to provide a logical subsystem organization separate from the physical organization.

Under this organization, one or more subsystems can be declared as default. This default designation permits general usage of the subsystems without explicit EU or subsystem assignment, as in COPY...TO DISK. Nondefault subsystems are assignable only by specific request, as in COPY...TO DISK (SUBSYSTEM 5).

Because there is only one disk directory for all of the EUs and subsystems, determining which EU or subsystem houses the file is not a concern in file retrieval. This organization

provides for great control and flexibility over the disk system and allows for both generic usage and specific file placement.

The introduction of 180-byte pack media occurred with the release of MCP1 and MCP2 (ASR 4.0 and 4.2). The larger sector size of pack media provided better utilization of the media, and the removable drives provided increased flexibility and lower costs.

The initial implementation provided what today are known as *unrestricted families*. Through specific naming schemes, assignment could be made to a specific pack, to a group of packs by way of a masked family name, or generically to any pack by the absence of a specific designation. In this way, packs could be organized and used like disk.

Restricted families were later introduced. These families required a specific family name designation in order to create or access files. Restricted pack families are equivalent to nondefault disk subsystems.

MCP/VS ASR 2.0 introduced the ID method for identifying disks and shared packs. For disks, the ID number replaced the EU number; for example: COPY...TO DISK (ID 7). This method was also used to generate lock addresses for shared disks and shared packs.

MCP/VS ASR 3.1 introduced a new version of packs (called *version 2*) to simplify this flexible but complicated organization of packs (called *version 1*). By streamlining the organization of pack families and eliminating redundant pack structures, version 2 packs offer a simpler organization and enhanced performance.

MCP/VS ASR 3.2 introduced another version of packs (called *version 3*) to enhance the performance of directory searches. By creating subsets (called *scramble strings*) of the entire directory, specific file name searches are limited to only part of the directory and results in enhanced performance. Directory blocking was also increased to reduce I/Os and to improve performance.

MCP/VS ASR 3.3 introduced a new version of packs (called *version 4*) with the ability to allow simultaneous access to a pack directory by multiple programs.

Version 1 pack families can be converted to version 2 or greater pack families, and all versions can coexist on the same pack subsystem. All the members of a family must be of the same version, however. Conversion and coexistence are discussed later in this section.

Caution

Data corruption results if all of the following conditions are true:

- A SCSI diskpack has a capacity greater than the limit enforced by the MCP
- The "available" tables have not been limited to the MCP limit by the use of the DISPKV utility, the ALTER NEW PACK command, or the ALTER FAMILY REBUILD command
- The MCP attempts to allocate an address beyond the limit enforced by the MCP

Components of Pack Subsystems

Pack subsystems consist of the following:

- The *pack* is the physical magnetic medium on which information is stored.
- The pack *unit* is also called a *drive* or *spindle.* The pack unit is considered a peripheral device.
- The *controller* handles the interface between the channel or data link processor (DLP) and one or more units. Some pack drives do not require a controller (for example, MD4s and MD8s); for these drives, the DLP assumes the function of the controller. Usually, a controller must be loaded with firmware. Loading can occur automatically during system initialization, or manually through the LH keyboard command or through the LOADFW standalone utility. For complete information on the LH keyboard command, refer to Volume 2; for a description of the LOADFW utility, refer to Section 12 in this volume.
- The *data link processor (DLP)* provides the interface between the host and the controller (or between the host and the unit for drives that do not require a controller). This interface is also known as a *channel.* Note that many pack types support the ability to connect several channels to a controller (or to a unit for packs that do not use a controller). In such a case, the channels are said to be *exchanged.* This technique provides multiple access paths between the host and the units, thus permitting simultaneous data transfer.
- The software controls the utilization of the components of the subsystem. The software provides the logical structure for the information stored on the pack subsystem.

The components of a pack subsystem are illustrated in Figure 31–1.

The unit is represented by a channel and unit address. This address is designated in this guide as *cc/uu.* The *cc* number, or channel designation, names a specific connection between the host and the controller. The *uu* number, or unit designation, names a specific connection between the controller and the unit.

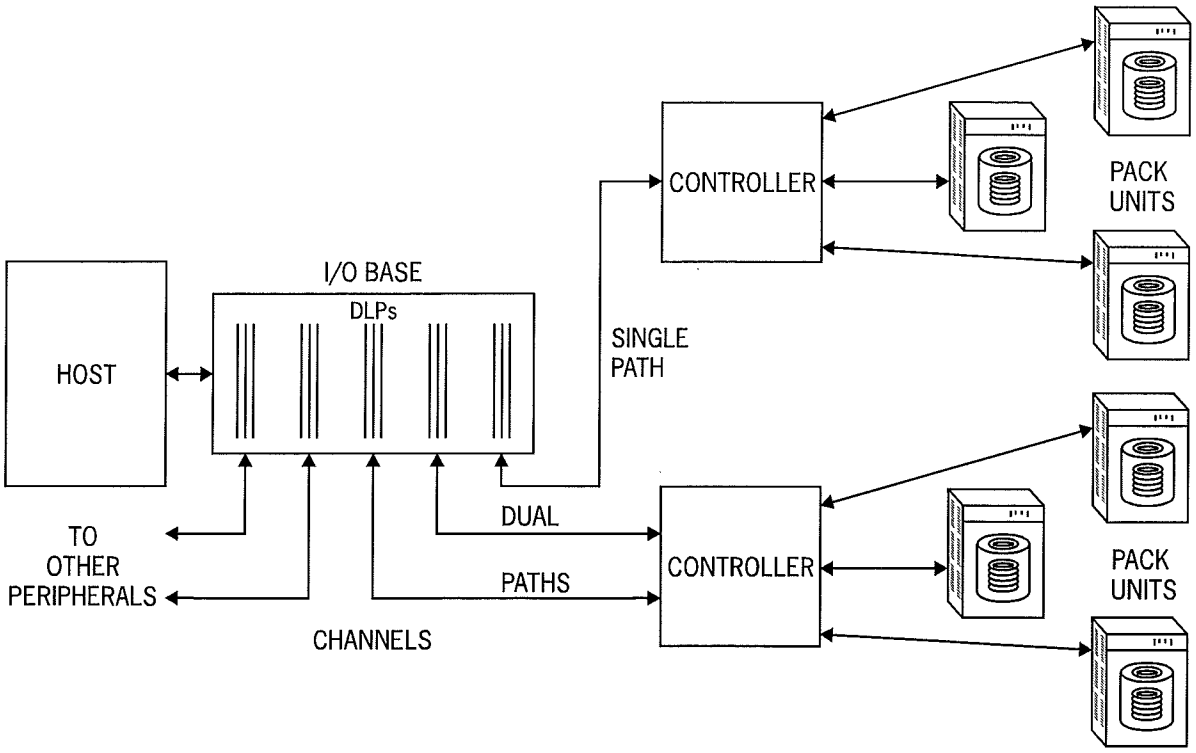


Figure 31–1. Components of a Pack Subsystem

Physical Organization of a Pack

Each pack consists of one or more *platters*: circular plates coated with a thin film of magnetic material. The platters are mounted on a central *spindle*. The spindle and platters may or may not be removable from the pack drive, depending on the pack model.

Associated with each platter is one or more read/write *heads* used to read and write information on the magnetic film. A head moves to the specified location on a platter to access data. The movement of the head to this location is called a *seek*.

Data is stored on a platter in concentric rings called *tracks*. All the tracks on the various platters of a pack that have the same radius form a *cylinder*. The controller identifies a track by the cylinder and head to which it belongs.

Tracks are divided into *sectors*. A sector is the smallest portion of physical data that can be uniquely addressed. On a pack there are 180 bytes per sector. The two methods for organizing sectors on a track are

- Sequential format
- Interlaced format

In sequential format, sectors on a track are arranged in sequential order: 1, 2, 3, 4, and so on. In interlaced format, which was often used by older models of packs, the sectors on a

track are arranged nonsequentially (for example, 1, 45, 2, 46, 3, 47, and so on). Figure 31–2 illustrates the two formats.

Sequential format offers improved data transfer time over interlaced format because adjacent sectors are contiguous and time is not spent waiting for the next logical sector to rotate. Interlace format packs offer increased capacity over sequential format packs because of smaller inter-sector gaps.

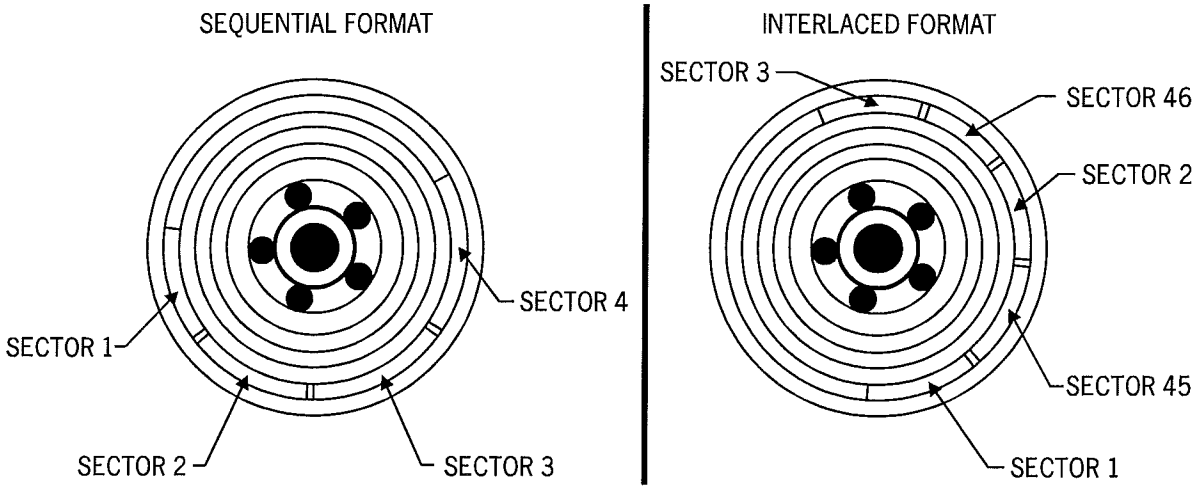


Figure 31–2. Sequential and Interlaced Organization of Sections

The physical organization of a pack is illustrated in Figures 31–3 and 31–4.

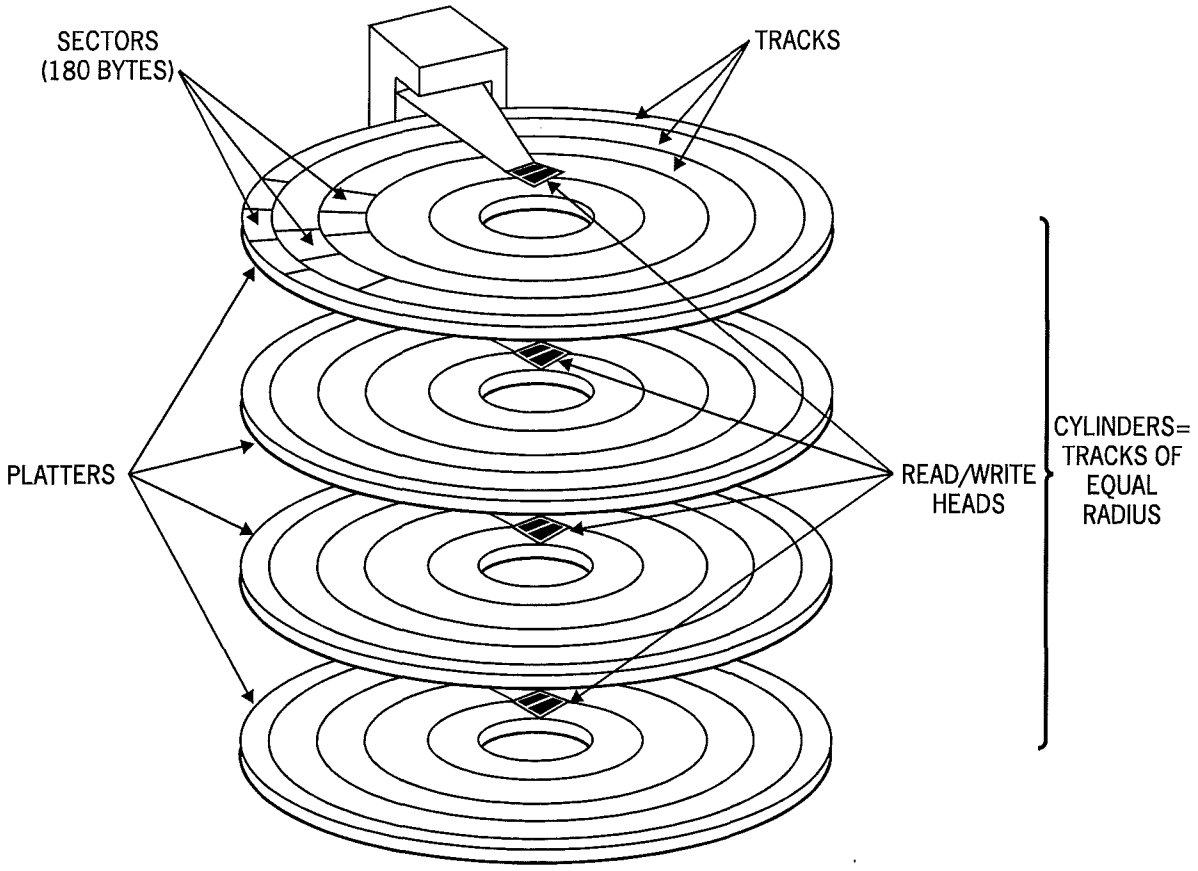


Figure 31–3. Physical Organization of a Pack (Side View)

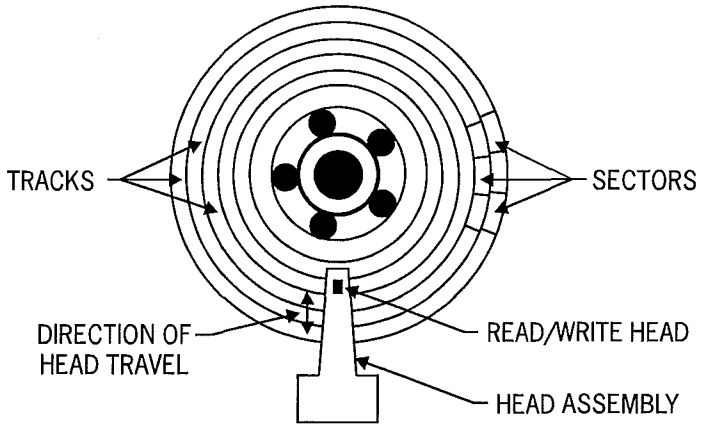


Figure 31–4. Physical Organization of a Pack (Top View)

For data access, the controller decodes each address into a cylinder/head/sector designation. This kind of designation can be seen on a Maintenance Log (MLOG). (For a complete description of the Maintenance Log, refer to the *V Series Systems Software Logging Operations Reference Manual.*) The heads are then moved to the specified cylinder. This action results in *seek time.* After the heads are positioned, the appropriate head is activated and must wait until the rotation of the platter brings the specified sector beneath it. This wait is called *rotational latency time.* Data transfer can then occur; the amount of time required for this data transfer is called *transfer time.* Additional time is also required for DLP and MCP queueing.

Organization of Pack Data

Data on a pack is organized into a logical hierarchy. This hierarchy—from lowest to highest—is as follows:

- Individual data is contained in a *field.*
- One or more fields compose a *record.*
- A given number of records compose a *block.* The number of records per block is called the *blocking factor.* The blocking factor is currently limited to 999.
- A given number of blocks are stored in an *area.* Physically, an area is made up of a given number of sectors. The sectors that make up an area have sequential addresses.
- One or more areas contain the contents of a *file.* A file is a named collection of related information. The number of areas per file is currently limited to 100.

Figure 31–5 compares the logical and physical organization of pack data.

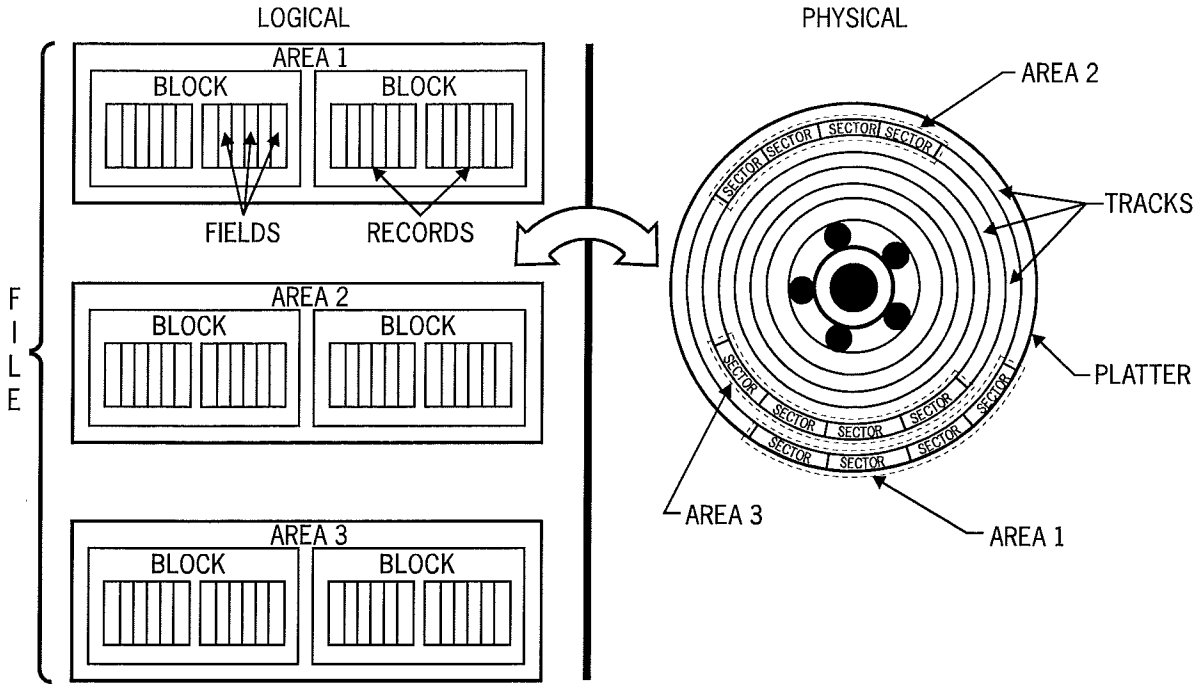


Figure 31–5. Logical Organization of Pack Data

The size of each field, the size and organization of a record, the number of records per block, the number of records per area, and the number of areas are all user defined.

Physically, a block begins on a sector boundary. If the block size (the record size multiplied by the blocking factor) is not equal to a whole number of sectors, dead space exists up to the next sector boundary (refer to Figure 31–6). Dead space cannot be used to store data and can result in space and access inefficiencies.

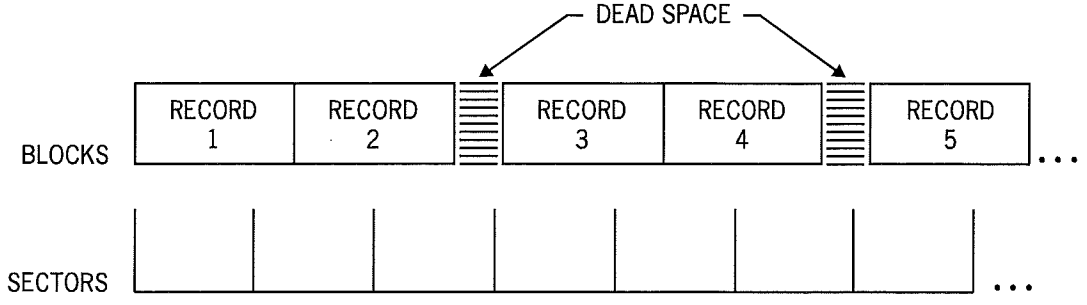


Figure 31–6. Physical Organization of Blocks

A program can open a file with a record size and a blocking factor that are different from those specified when the file was created. Typically, the blocking factor is increased to

improve I/O efficiency. Great care must be taken to ensure that proper record and block alignment occur and that dead space is not accessed as data.

Logical Organization of Packs

Each physical pack belongs to a pack *family*. A family is a logical grouping of one or more packs. The MCP logically links family members together so that they are treated as a single entity for storing files (refer to Figure 31–7). Therefore, packs can be added to a pack family as space requirements increase. When a pack is labeled, it is assigned to a pack family by specifying the pack family name. A pack family can contain packs of different pack types, sector formats (sequential or interlaced), and channel and unit designations. The link between packs in a pack family is solely a logical link as indicated by a common family name.

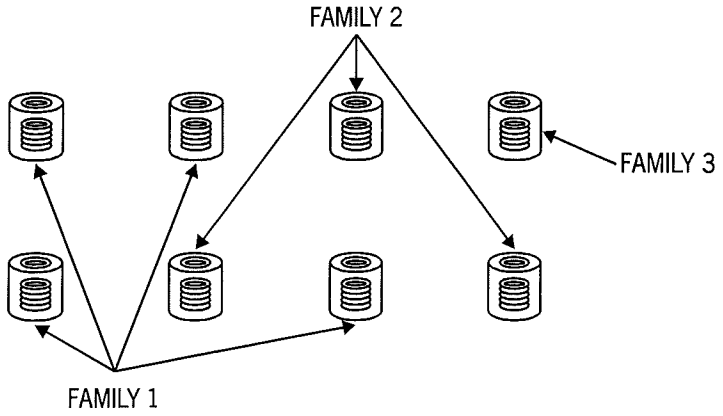


Figure 31–7. Logical Organization of Packs

Physical and Logical Organization of Files

A file is stored in a number of areas. Each area is a group of contiguous sectors; however, the areas in which a file is stored are not necessarily contiguous. All the areas of a file are the same size, as specified when the file is created. Different files can have different area sizes.

A file can be stored in one or more areas (up to 100). The MCP allocates areas for the file as needed and as available. Thus, the MCP may place the various areas of a file on different packs. These packs are always members of the same pack family. In other words, a file can be physically spread across a pack family.

This organization of packs and files is illustrated in Figure 31–8.

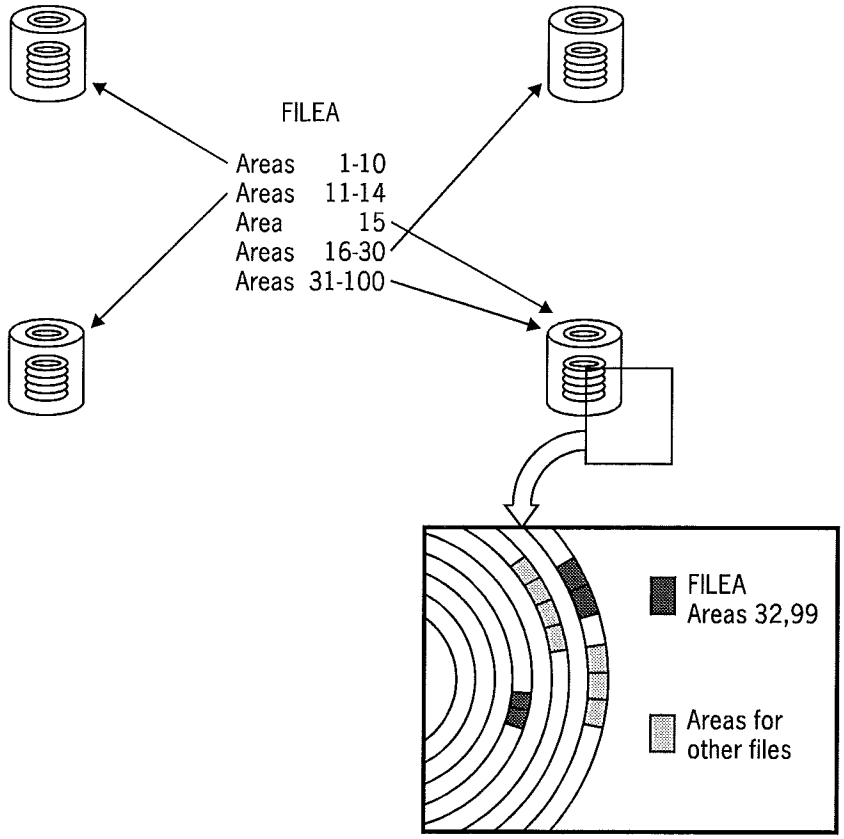


Figure 31–8. Organization of Packs and Files

Logically, the MCP keeps track of the file and of the physical addresses of each area of the file in special structures: a *directory* and *header*. These special structures are stored within the same pack family as the file.

A directory contains file names, pointers to headers and other information.

Each file has a corresponding file header that contains the physical addresses of the areas of the file. The header also contains other information such as the length of the records in the file. The MCP accesses each area of a file by obtaining its physical address from the file header.

This organization of directories and headers is illustrated in Figure 31–9.

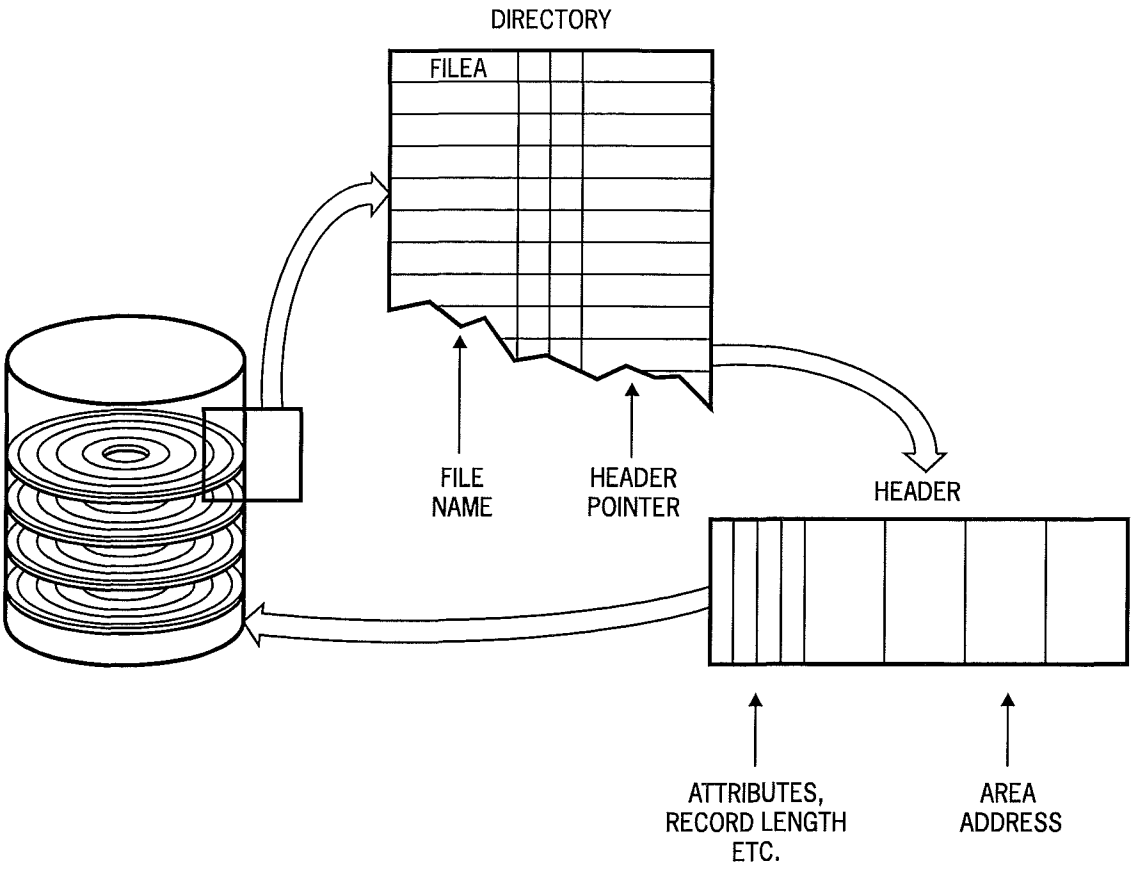


Figure 31–9. Organization of Directories and Headers

MCP Structures

In addition to the directory and header, there are three other MCP structures on each pack:

- The *label*
- The *master available table*
- The *available table*

The label contains the pack name and pointers to the master available table, the available table, and the directory.

The master available table contains address/length pairs for all usable areas on the pack. The master available table is created when the pack label is created, and changes when the pack is reinitialized, or when an XP command or RXP command changes the usable space on the pack. For more information on the XP and RXP commands, refer to Volume 2.

The available table contains address/length pairs for all usable and currently unused space on the pack. This table is dynamic and changes as the MCP utilizes the pack for storage.

When the pack structures are first built, the available table is virtually identical to the master available table.

Pack Versions

The organization of a pack family, the presence or absence of a directory or headers on a pack, and the way in which the MCP accesses files varies depending on the version of the pack. The pack versions are as follows:

- Version 1
- Version 2 (introduced at ASR 3.1)
- Version 3 (introduced at ASR 3.2)
- Version 4 (introduced at ASR 3.3)

Different methods for creating MCP structures are required for version 1 packs as opposed to version 2 and greater packs.

Organization of Version 1 Pack Families

For a discussion of the development of version 1 pack families, refer to "Development of Unisys Disk and Pack Subsystems" earlier in this section.

Types of Version 1 Packs

There are three types of version 1 packs:

- Master
- Base
- Continuation

The use of these pack types is based on the concept of a central, logical access point to the pack family for the pack subsystem (refer to Figure 31–10).

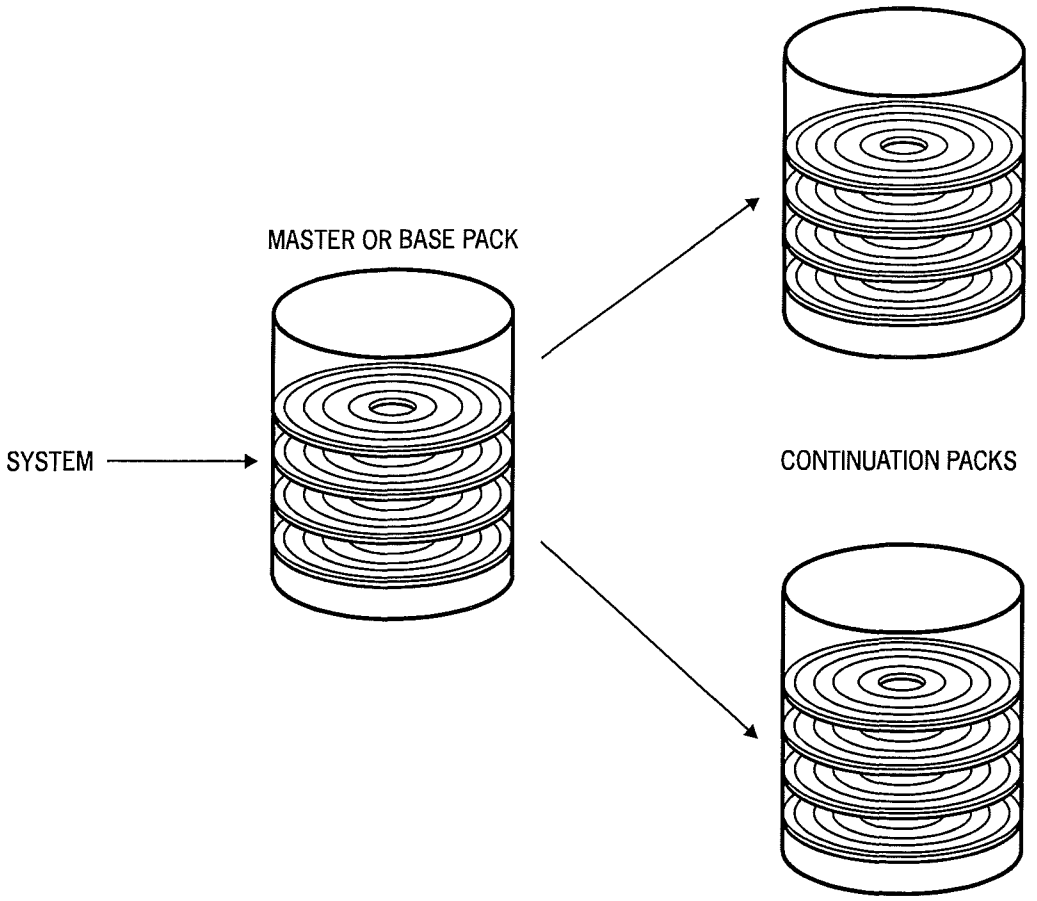


Figure 31–10. Access Point to a Version 1 Pack Family

A master pack is explicitly declared when the pack is labeled. A pack that is not declared as a master pack is considered to be a continuation pack. The status of a pack as a master or continuation pack is a physical attribute that is specified during the labeling procedure.

A base pack can be thought of as a logical master pack. When a file is created and an explicitly declared master pack has not been declared (or when the master pack is for some reason unavailable), the MCP selects a pack to be the base pack and to serve the same function that a master pack would. All master packs are base packs, but not all base packs are master packs.

A base pack is the primary point of access for the version 1 pack family. In addition to the label, master available table, and available table, each base pack contains the following MCP structures:

- A directory which contains an entry for each file in the pack family
- A header for each file in the pack family

In addition, the base pack contains the initial area of each file in the pack family.

The MCP uses continuation packs to provide additional storage capacity for the pack family. A continuation pack can contain one or more areas of a file. Although the initial

area of the file must be on the master or base pack, the remaining areas can be spread across the pack family. The MCP normally chooses a pack on which to place each area as it is allocated. The MCP bases this choice on several criteria, although the pack with the greatest percentage of available space is usually the prime candidate to receive the area.

In addition to a label, master available table, and available table, each continuation pack contains a directory and a header for any file that has one or more areas on that pack. Therefore, in a version 1 pack family, all base and continuation packs contain a directory and headers.

Types of Version 1 Families

There are two types of version 1 pack families:

- Restricted
- Unrestricted (also known as a *system resource pack*)

A pack is explicitly declared as restricted or unrestricted when the pack is labeled.

A restricted version 1 pack family can have only one master pack, but does not require any. An unrestricted version 1 pack family can have multiple master packs, but does not require any.

Both restricted and unrestricted version 1 pack families can have multiple base packs (that is, the MCP can assign more than one base pack if no master pack has been declared).

Each family (restricted or unrestricted) has a *family name*, which can be up to 6 characters long. The family name is used as follows:

- A restricted version 1 pack family requires that the exact family name be used for creating or accessing a file within the family.
- An unrestricted version 1 pack family permits you to use an exact, masked, or null family name for creating or accessing a file within the family. Several MCP commands and utility functions access unrestricted families by using the keyword PACK (for example, COPY...TO PACK). Family name masking is the ability to specify a wildcard as part of the family name; for example, specifying *A* = selects all unrestricted family names beginning with A.

How the MCP Accesses Files on a Version 1 Pack Family

Figure 31–11 illustrates a typical configuration of a version 1 pack family.

In this example, 24/0 is the master (and base) pack for this pack family. It contains a directory that points to the header for each file stored on the family. In addition, this pack contains the initial area of each file stored on the family.

This pack family also consists of two continuation packs on 24/1 and 24/2. Each continuation pack also contains a directory with an entry for each file that has an area on the pack. Each entry points to the header for each file.

The file FILEA is stored on all three disks: the initial area is stored on 24/0, and the remaining areas are spread across the two continuation packs. These areas were assigned by the MCP on an as-needed and as-available basis.

When you open FILEA, the MCP accesses the directory on the master pack, and then locates the header for the file from the directory. When a record is read or written, the MCP calculates which area of the file contains the record and accesses that area address in the header. The area address can be in one of the two following formats:

- If the area is on the master pack, the area pointer contains the physical address of the area.
- If the area is on a continuation pack, the area pointer contains a continuation indicator and the serial number of that continuation pack. The MCP must access the directory on that pack, locate the file, access the local header, and obtain the physical address of the area.

In actual practice, this process of locating and resolving area addresses is optimized to improve efficiency.

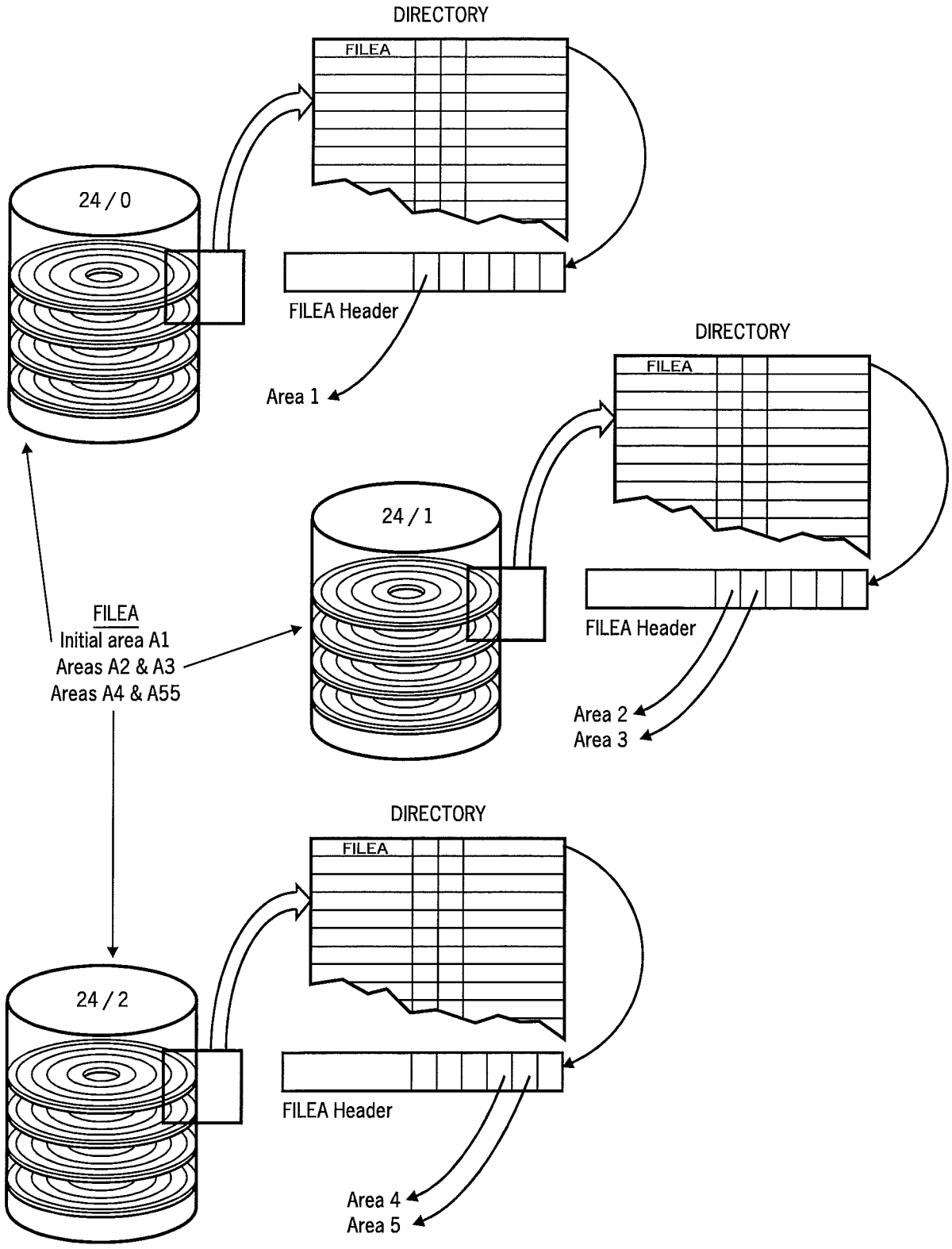


Figure 31–11. Version 1 Pack Family Configuration—Example 1

Figure 31–12 illustrates another typical configuration of a version 1 pack family.

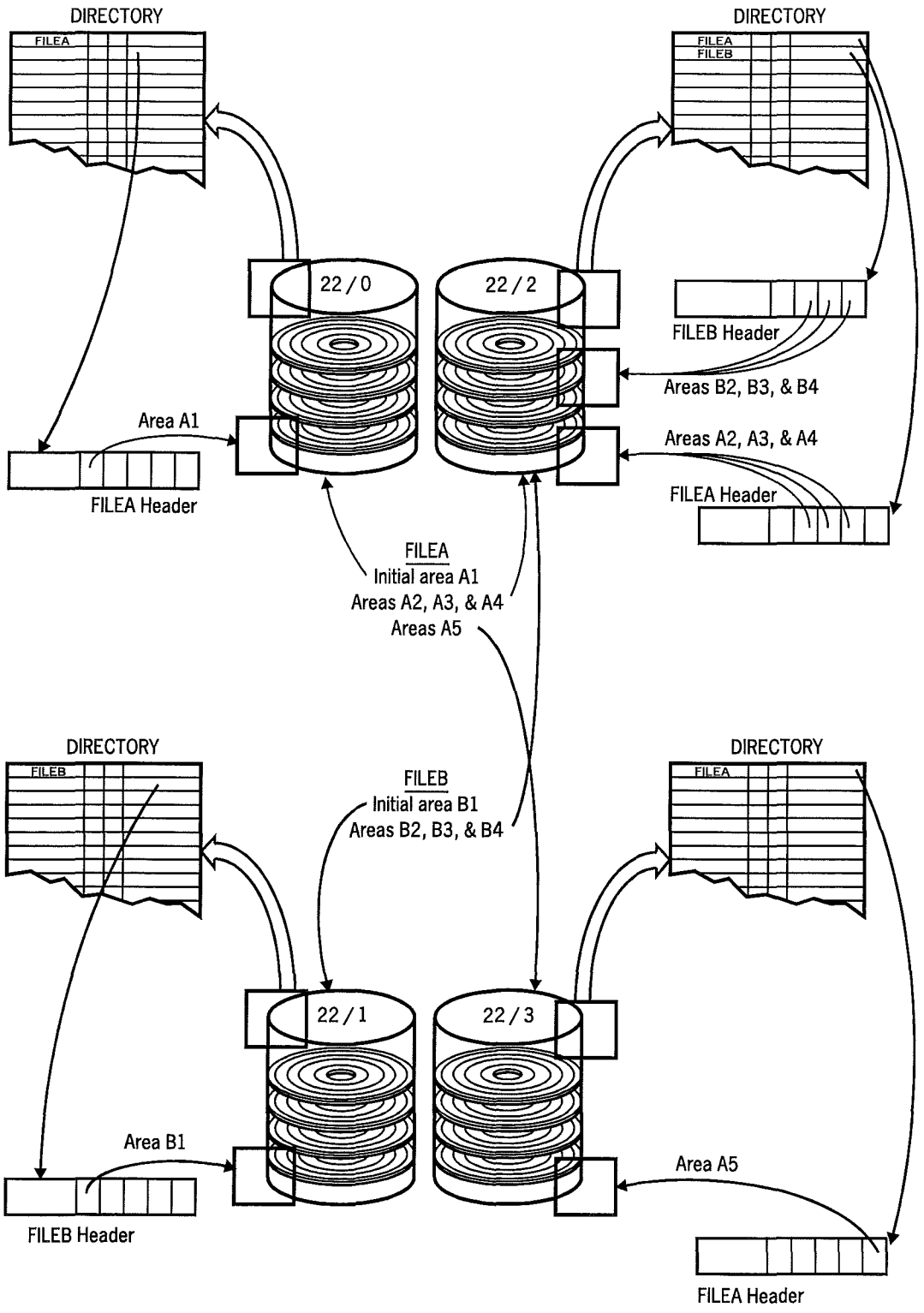


Figure 31–12. Version 1 Pack Family Configuration—Example 2

In this example, there is no declared master pack for this family. Two files have been created: FILEA and FILEB. When FILEA was created, the system chose 22/0 as the base pack and created a directory entry, a header, and the initial area for the file on this pack. (In the absence of a master pack, the system usually chooses as a base pack the pack with the greatest percentage of space available.) When FILEB was created, the system chose 22/1 as its base pack.

This pack family also consists of two nonbase continuation packs on 22/2 and 22/3. Each continuation pack contains a directory with an entry for each file that has an area on the pack. Each entry points to the header for that file, which contains appropriate area addresses.

File areas are allocated as needed, usually on the pack having the greatest percentage of space available at the time of the request. The exception to this is the initial area assignment; the initial area always goes to the base pack of the file. FILEA is stored on three packs: the initial area is stored on the base pack 22/0, and the remaining areas are spread across the continuation packs 22/2 and 22/3. FILEB is stored on two packs: the initial area is stored on the base pack 22/1, and the remaining areas are stored on the pack 22/2.

When FILEA is opened, the MCP accesses the directory on the base pack for the file and then locates the header for the file from the directory. When a record in FILEA is read or written, the MCP calculates which area of the file contains the record and accesses the appropriate pack.

Labeling a Version 1 Pack

When a version 1 pack is labeled, you declare the pack as one of the following:

- A restricted pack
- A restricted master pack
- An unrestricted master pack
- An unrestricted pack

You can label a version 1 pack by using the DISPKV utility (refer to Section 3 of this volume). You can use the ALTER PACK command to modify an existing label (refer to Volume 2).

Building MCP Structures on a Version 1 Pack

To build pack-resident MCP structures on a version 1 pack, you must use the DISPKV utility.

Rebuilding MCP Structures on a Version 1 Pack

To rebuild MCP structures after an initialization or when a catastrophic failure occurs, you must use the DISPKV INITIALIZE, CONFIGURE, or RECONFIGURE commands.

Purging All Files from a Version 1 Family

To remove individual files or a group of files from a pack, use the RM keyboard command.

To remove all files from a pack, use the DISPKV RECONFIGURE command. This command removes all files by reinitializing the directory and available tables. You can also remove all files from a pack by using the RM keyboard command and a completely masked file name. However, this process requires a null mix for the MCP and can take considerable time to complete.

Renaming a Version 1 Pack or Pack Family

To rename a pack, use the DISPKV RENAME command or the ALTER PACK keyboard command. To rename an entire pack family, you must rename each member separately.

Organization of Version 2 and Greater Pack Families

The organization of version 2 and greater pack families alleviates the confusion and inefficiencies caused by multiple directories, headers, and base packs. Version 2 and greater pack families differ from version 1 pack families in the following ways:

- There is a single base pack for each family. The use of an explicitly declared master pack (or implicit base packs) has been replaced by the use of a single, required, explicitly-declared base pack that acts as the access point for the pack family.
- There is a single directory for each family. This directory is located only on the base pack.
- There is a single header for each file in the family. This header is located only on the base pack.
- The initial area of a file can be stored on any member of the pack family.
- There is one, and only one, resource (unrestricted) family.
- The ALTER PACK and ALTER FAMILY keyboard commands assume many of the functions of the DISPKV utility for labeling packs and building MCP structures. For a complete description of the ALTER command, refer to Volume 2.

The directory for version 3 and greater pack families differs from the directories for version 1 and version 2 pack families. The directory for a version 3 or greater pack consists of 10 subsets of the entire directory. To locate a specific file name, only one of the subsets is searched rather than the entire directory.

Types of Version 2 and Greater Packs

There are two types of version 2 and greater packs:

- *Base*
- *Continuation*

A base pack is the primary point of access for version 2 or greater pack families. There is one and only one base pack for each version 2 or greater pack family. The base pack is declared by the user when the pack is labeled. In addition to a label, a master available table, and an available table, the base pack contains the following MCP structures:

- A directory, which contains an entry for each file in the pack family
- A header for each file in the pack family

A continuation pack is any member of the pack family that has been declared as a continuation pack. Continuation packs are used to provide additional storage capacity for the pack family.

The only MCP structures contained on version 2 and greater continuation packs are the label, the master available table, and the available table; version 2 and greater continuation packs do not contain directory or header structures. Therefore, there is one and only one directory for the pack family. Also, there is one and only one header for each file in the family.

Types of Version 2 and Greater Pack Families

There are two types of version 2 and greater pack families:

- *Restricted*
- *Resource* (also called *unrestricted* or *default*)

A restricted version 2 or greater pack family requires that you use the exact family name when accessing a file within the family.

There can be only one version 2 or greater resource pack family at any given time. This resource family requires that you use the exact family name or a null family name when accessing a file within the family. Several MCP commands and utility functions access the resource family by specifying the keyword PACK (for example, COPY...TO PACK).

Family name masking was allowed for version 1 resource packs (refer to the description earlier in this section), but because there can be only one version 2 or greater resource pack family, family name masking is ignored. Commands and file open requests containing family name masking are directed to the resource family regardless of name matching.

Restrictions

A version 2 or greater pack family cannot have more than 15 members (one base and 14 continuation packs).

Foreign or interchange mode packs are no longer supported.

Certain system files that are accessed by utilities using a channel/unit designation (for example, MCP and configuration files accessed by LOADER and firmware files accessed by LOADFW) must reside on the base pack of the family.

When you use the SQP keyboard command for a version 2 or greater continuation pack, the system requires exclusive access to the base pack of the family in order to access the directory.

The CHANGE and REMOVE commands which contain a channel/unit designation must specify the base pack of the family.

Once a file has been opened on a family, OU commands are not allowed to direct areas to other families.

How the MCP Accesses Files on a Version 2 or Greater Pack Family

Figure 31–13 illustrates a typical configuration of a version 2 or greater pack family.

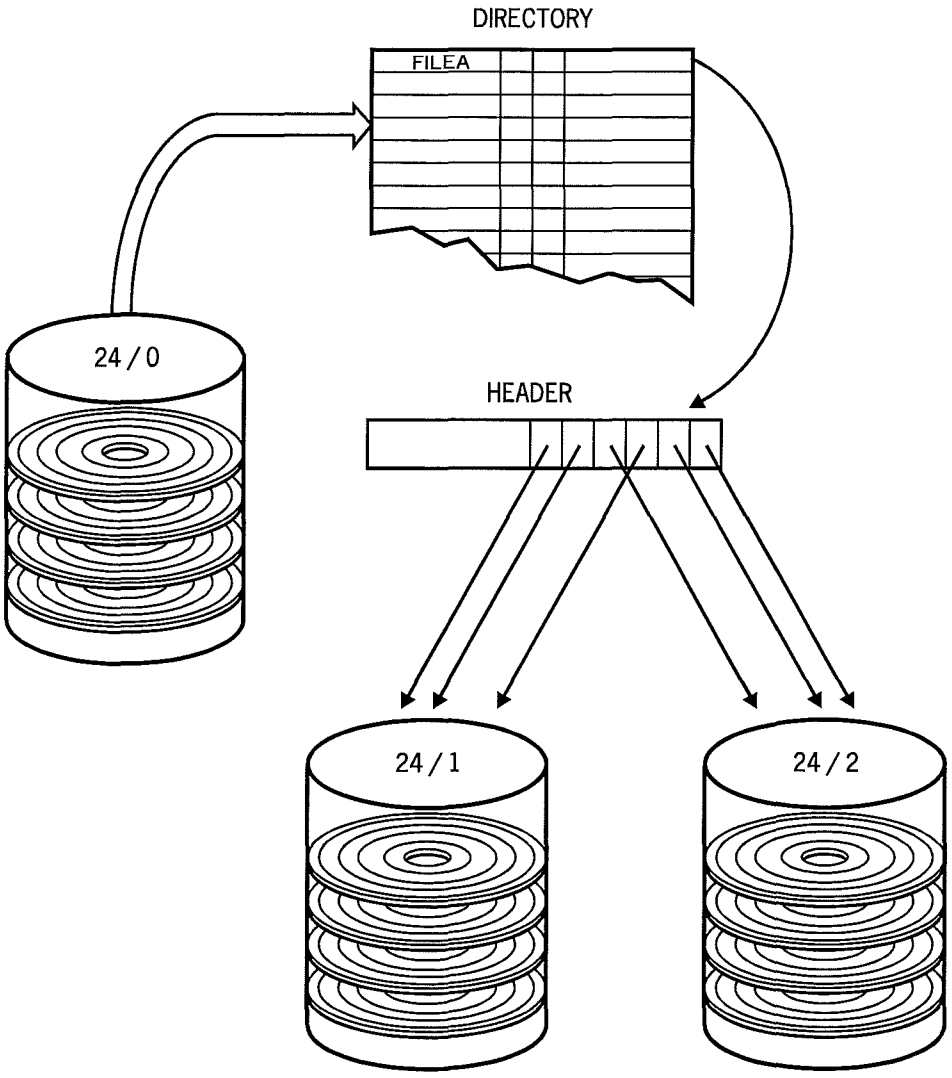


Figure 31–13. Configuration of a Version 2 or Greater Pack Family

In this example, 24/0 is the base pack for this restricted pack family. Pack 24/0 contains the directory that points to the header for each file stored in the family. The areas of FILEA are stored on the continuation packs, 24/1 and 24/2. The initial area of FILEA is on the continuation pack 24/1.

When FILEA is opened, the MCP accesses the directory on the base pack and locates the file header from the directory entry. The MCP then locates the address of each area of the file from the header.

How the MCP Accesses a Directory on a Version 3 or Greater Pack Family

Packs of version 3 and greater are organized exactly the same as version 2 packs. The difference between these versions is in the way the directory is built and accessed. Prior to version 3 packs, the directory was searched from the beginning until a file name was found. With version 3 and greater packs, only a fraction of the directory must be searched.

Scramble Strings

A version 3 or greater pack directory consists of 10 subsets of the complete directory. These subsets are called *scramble strings*. A file name is "scrambled" into a single number from 0 to 9 and is placed into the scramble string corresponding to that number. The file names are scrambled in a manner that attempts to create an even distribution of file names in each scramble string.

Figure 31–14 illustrates a directory structure for a version 3 or greater base pack. The starting scramble string address when searching for a specific file name is as follows:

```
directory-address + (scramble-string number * directory-blocking-factor)
```

The end of each scramble string is forward-linked to the beginning of the next scramble string; thus, a search of the entire directory remains functional (for example, PD A=).

The following are the special characters used in Figure 31–14.

- # This character indicates the "first-sector-for-scramble," which is the beginning of each scramble string (specific file name searches start here).
- & This character indicates the "last-logical-scramble-sector," which is the last sector that contains assigned file entries (specific file name searches end here).
- * This character indicates the "last-physical-scramble-sector," which is the physical end of each scramble string.

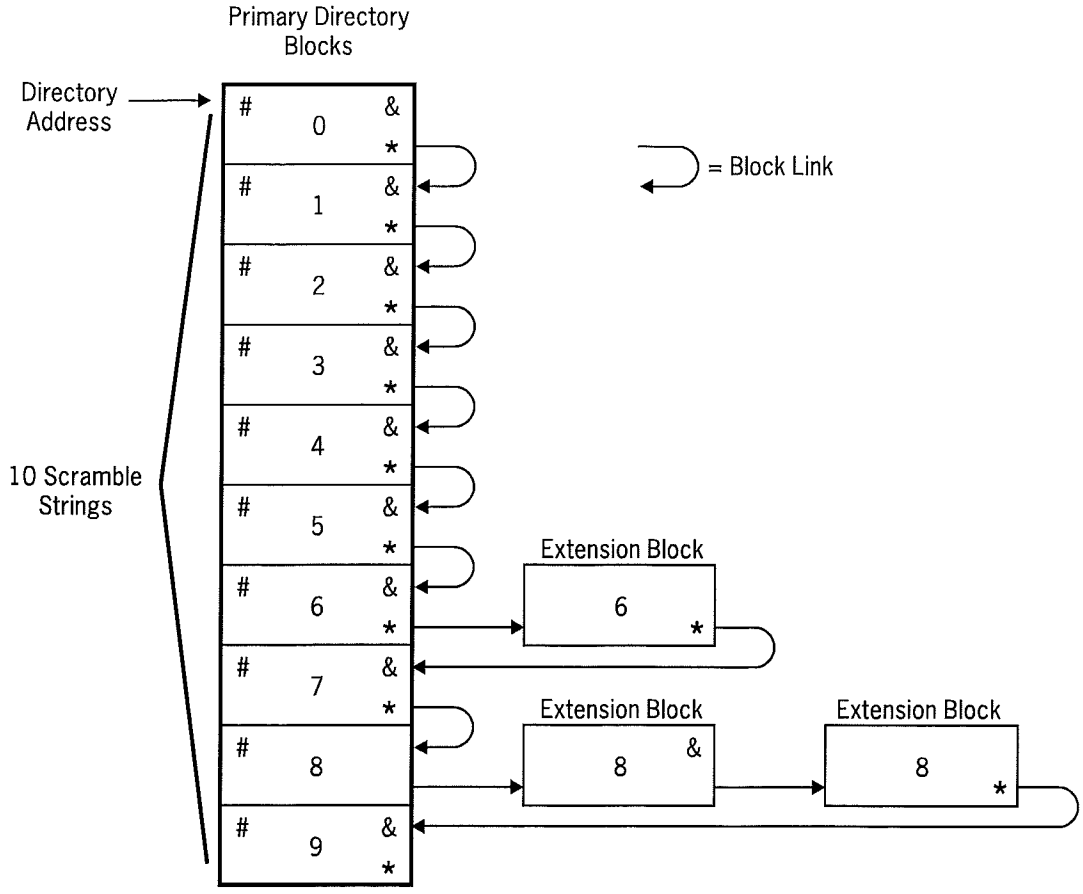


Figure 31–14. Directory Structure—Version 3 or Greater Base Pack

Directory Blocking

Each block of the version 3 or greater pack directory contains 20 records, as opposed to version 1 and version 2 directories, which contain 10 records for each block. Each record contains 11 file names along with other information. When attempting to locate a file name, the MCP reads in one block at a time, so the larger block size can decrease the number of reads that are necessary to locate a file name.

Directory Searches for Specific Filenames

A search of the entire directory is not needed to locate a specific file name. Instead, a search is limited to only one scramble string because the file name is scrambled as explained previously. The result is a decrease in search time to locate a file.

Directory Searches for Masked File Names

A search of the entire directory must still be performed for masked file names (such as PD A=) because it is not known what scramble string a masked file name belongs to.

These searches are slightly more efficient than version 1 and version 2 pack searches because of the larger block size, as explained previously.

How the MCP Accesses a Directory on a Version 4 Pack Family

The structures on version 4 packs are organized identically to version 3 packs. The difference between version 3 and version 4 packs is the directory access method. For version 3 packs, only one directory access is allowed at a time by requiring that address 20 be locked before the directory is searched. For version 4 packs, up to 10 directory accesses are allowed simultaneously by locking the primary scramble string blocks independently. This allows, for example, a total of 10 file opens and closes to occur simultaneously. (Certain directory accesses must still be single-threaded to avoid deadly embrace situations.)

Version 4 Locking Schemes

Scramble strings 0 through 9 are locked and unlocked by their primary block address. Currently, these addresses are 20, 40, 60, 80, 100, 120, 140, 160, 180, and 200 for scramble strings 0 through 9 respectively.

The locking schemes for accomplishing specific operations are described in the following sections.

Opening and Closing Files

Opening a file requires a lock and unlock of the primary block address of the scramble string where a file resides. This address is saved in the soft IOAT (input-output assignment table) of the file during the file open process. During the file close process, the saved address is used to lock and unlock the affected scramble string.

Specific File Name Changes

The primary block address of scramble string 0 is locked in order to single-thread the function of changing a specific file name. A maximum of two additional primary block addresses of scramble strings are also locked (one for the current file name and one for the new file name). The primary block address of each affected scramble string is unlocked in descending numerical order.

Masked File Name Changes

The primary block addresses of scramble strings 0 through 9 are locked in ascending numerical order before a masked file name change is performed, because all of the scramble strings will probably be accessed to complete this function. The primary block address of each scramble strings is unlocked in descending numerical order (9 through 0).

Specific File Name Removes

During the operation of removing a specific file name, the primary block address of the scramble string where a file resides is locked and unlocked.

Masked File Name Removes

First, to single-thread the function of a masked file-name remove, the primary block address of scramble string 0 is locked, and it is kept locked until the function is complete. The primary block addresses of scramble strings 1 through 9 are locked and unlocked in sequence as each scramble string is searched.

Miscellaneous Directory Accesses

For any other function, such as pack recovery, that accesses the directory and requires the directory to be locked, the locking is performed per the primary block address of each scramble string, rather than by locking address 20.

BLT Entries

If a LIMIT BLT (Limit Block Lockout Table) record is included in a configuration file, it should specify a minimum of 40 block lockout entries. This is because up to 10 scramble strings and file headers can be locked at the same time, resulting in 20 entries in the BLT. Any additional BLT entries that are required are then available for utilization by user programs or other MCP functions.

If a LIMIT BLT record is not included in a configuration file, the number of entries is calculated by the MCP and will always exceed the minimum requirement of 40. For information on the LIMIT BLT record, refer to *Volume 1*.

Shared Version 4 Packs

Since the locking protocol for version 4 packs is different from previous versions, if packs are shared between multiple systems, an MCP release level of at least 3.30 must be running on each system.

In order to convert a shared pack family to version 4 format, the following steps must be taken in the specified order:

1. All packs in a family must be saved on each system that they are declared on.
2. A family should be altered to version 4 format on only one system using the ALTER syntax as explained later in this document.
3. Ready the previously saved packs on all applicable systems.

Using the ALTER PACK, ALTER NEW PACK, and ALTER FAMILY Commands on Version 2 and Greater Packs

The ALTER PACK, ALTER NEW PACK, and ALTER FAMILY commands are used as described in the following paragraphs.

Using the FILES Attribute to Preallocate Additional Directory Blocks

One of the options of the ALTER NEW and the ALTER FAMILY commands is the FILES attribute (see the description of the ALTER command in Volume 2 for details). This option performs the following functions:

- On version 2 packs it preallocates contiguous directory blocks at the end of the directory in addition to the default number of directory blocks.
- On version 3 and greater packs it preallocates contiguous directory blocks for each scramble string in addition to the default number of directory blocks.

If you know approximately how many files will reside on a pack, you can use the FILES attribute to further reduce the search time for a file.

Notes: *It is advantageous to preallocate additional contiguous directory blocks to minimize head movement if you know that more than the default number of directory blocks will be required; since any directory blocks dynamically allocated by the operating system will not be contiguous to the initial blocks.*

Any preallocated directory blocks are not searched until they are used, so allocating additional directory blocks will not affect system performance.

Labeling and Building MCP Structures

You must use the ALTER NEW PACK command to label a version 2 or greater pack. For information on the ALTER NEW PACK command, refer to Volume 2. Options for this command enable you to specify whether the pack is base or continuation and whether it is restricted or resource.

Rebuilding MCP Structures

To rebuild MCP structures after an initialization or when a catastrophic failure occurs, you can

- Use the ALTER NEW PACK command for each member of the pack family.
- Use the ALTER FAMILY REBUILD command for the entire pack family.

These commands reinitialize all MCP structures, purging all files and areas marked out by the XP command in the process.

An ALTER NEW PACK command that specifies a continuation pack assigns the next available family index to the pack (up to the limit for version 2 and greater of 15 packs per family).

The ALTER FAMILY REBUILD command reestablishes the existing base pack as family index 1, and reassigns family indexes to each continuation pack sequentially.

Note: *You can use the ALTER NEW PACK command only once for a particular continuation pack. Each ALTER NEW PACK command causes the family index to increment by one. As a result, after use of this command, family indexes will be out of sequence, and the number of packs that can be declared for the family will be fewer than 15 packs.*

To get the indexes back in sequence, you have the following choices:

Perform an ALTER NEW PACK command on the base pack and on each continuation pack in the family.

Use the ALTER FAMILY REBUILD command

Use the INDEX option of the ALTER NEW PACK command.

Purging All Files from a Pack Family

To remove individual files, use the RM keyboard command. To remove all files from a pack family, use the ALTER FAMILY PURGE keyboard command.

Renaming a Pack or Pack Family

To rename an individual pack, use the ALTER PACK keyboard command. To rename all the members of a pack family, use the ALTER FAMILY RENAME command.

Comparison of Pack Versions

Table 31–2 compares the features of version 1 with the features of version 2 and greater packs.

Table 31–2. Comparison of Pack Versions

Feature	Version 1	Version 2 and Greater
Number of master packs per family	Unrestricted: 0 through n Restricted: 0 or 1	1
Multiple base packs possible per family?	Yes	No
Continuation packs possible per family?	Limited only by configuration maximums	14
Multiple resource families?	Yes	No
Directory and headers on continuation packs?	Yes	No
Multiple simultaneous directory accesses?	No	Version 4 only
Family name masking for resource family?	Yes	Directed to resource family regardless of name
Utility or command to generate label	DISPKV	ALTER NEW PACK
Utility or command to alter label	DISPKV or ALTER PACK	ALTER PACK
Utility or command to generate directory and available tables	DISPKV	ALTER NEW PACK
Default directory size	110 files	1100 files (Version 2) 2200 files (Version 3 and greater)
Utility or command to rebuild MCP structures	DISPKV RECONFIGURE for each pack	ALTER NEW PACK for each pack or ALTER FAMILY REBUILD for entire family
Utility or command to purge all files from family	DISPKV RECONFIGURE for each pack	ALTER FAMILY PURGE for family
Utility or command to rename pack or family	DISPKV RENAME or ALTER PACK for each pack	ALTER PACK for each pack or ALTER FAMILY RENAME for family

Converting Version 1 Families to Version 2 or Greater Families

Perform the following steps to convert an existing version 1 family to a version 2 or greater family. For additional important information on converting shared packs, refer to "Shared Version 4 Packs" in this section.

1. Dump all the desired files from the version 1 family to tape or other medium using SYSTEM/COPY or other such utility. As subsequent steps will destroy the original information, use of the COMPARE options and careful handling are highly recommended.
2. Determine any bad areas on each pack by using the WXP keyboard command, or by using DISPKV or PTD to reinitialize or reconfigure each pack.
3. Convert the family to the version 2 or greater format. This can be accomplished by converting the entire family at once or by converting each family member individually.
   a. To convert the entire family at once, follow these steps:

      Save each member of the family.

      Use the ALTER FAMILY REBUILD keyboard command. The MCP will automatically assign the base pack for the family (usually the previous master pack) and assign all other packs found with the same family name as continuation packs. Note that when converting a Version 1 pack with the ALTER FAMILY REBUILD command, if you do not specify a version to convert to, the default is to convert to Version 3.

      Ready the base pack and then each of the continuation packs. Refer to Example 1 for conversion syntax.

   b. To convert packs individually, use the following procedure:

      Save the pack that will be designated as the base pack of the family.

      Delete all other members of the family with the DL keyboard command.

      Create the base pack with the ALTER NEW PACK keyboard command.

      Ready the base pack.

      Add in each additional continuation pack with the PACK keyboard command. Use the SAVED attribute to avoid any pack version conflicts.

      Ready each of the continuation packs. Refer to Example 2 for conversion syntax.
4. Use the XP keyboard command to delete any bad areas on any pack as necessary.
5. Reload the files previously dumped.

Note: *The ALTER FAMILY & ALTER PACK keyboard commands allow attributes to be specified to override the existing attributes of the family or pack. Refer to System Operations Guide, Volume 2 for complete syntax and descriptions of the commands.*

You must be able to save a pack in order to use ALTER NEW PACK to convert it. You cannot save a pack in use by the MCP, such as a pack designated for the system dump file (USE DUMP PACK...). Thus, converting all pack families may require advanced planning.

Example 1:

```
Old family - 24/0 MCPACK (Restricted, Master)
             24/1 MCPACK (Restricted, Continuation)
             24/1 MCPACK (Restricted, Continuation)

SV 24/0, 24/1, 24/2
ALTER FAMILY MCPACK REBUILD
RY 24/0
RY 24/1, 24/2
```

Example 2:

```
Old Family - 24/0 BAPACK (Restricted, Master)
             24/1 BAPACK (Restricted, Continuation)
             24/2 BAPACK (Restricted, Continuation)


SV 24/0
DL 24/1, 24/2
ALTER NEW PACK 24/0
RY 24/0
PACK 24/1 SAVED
PACK 24/2 SAVED
ALTER NEW PACK 24/1
ALTER NEW PACK 24/2
RY 24/1, 24/2
```

If You Use System Security with a Mirrored User File

Special procedures are necessary for converting the diskpack that contains the mirrored security accesscode file, known as the user file or USERFL.

System security is invoked by the SECURITY record in the system configuration file. The mirrored USERFL is created by the MEDIA phrase in the SECURITY record.

Use the following procedure to convert the diskpack family that is named in the MEDIA phrase of the SECURITY record. This procedure can be used for converting between any two diskpack versions.

1. Perform step 1 of the procedure "Converting Version 1 Families to Version 2 or Greater Families." In this step you dump the files from the diskpack to tape. When you do this step for the diskpack that contains the mirrored user file, do not dump the

mirrored user file. Use the EXCEPT syntax of the SYSTEM/COPY utility to accomplish this.

The name of the mirrored user file is .USR*n*L where *n* is the system number. An example of the SYSTEM/COPY syntax for system number 1 could be:

```
COPY = EXCEPT (.USR1L) FROM PACK1 TO TAPE1
```

2. Perform steps 2, 3, and 4 of the procedure "Converting Version 1 Families to Version 2 or Greater Families." to convert the diskpack family to the new version.
3. To create a new mirrored user file on the newly converted diskpack family, halt/load the system.
4. Reload the files previously dumped.

Converting Between Version 3 and Version 4 Packs and Families

Version 4 packs can be created from version 3 packs and vice versa without rebuilding the pack directory. Only the pack label and EU table entry need to be changed, to reflect the version of the pack. For this reason file information on the pack is preserved.

The ALTER PACK command can be used to create a version 4 pack from a version 3 pack as follows:

```
ALTER PACK <cc/u> VERSION 4
```

Any other parameters that are currently allowed on the ALTER PACK command can also be specified.

The ALTER FAMILY command can be used to create a version 4 pack family from a version 3 pack family as follows:

```
ALTER FAMILY <family name> RENAME NAME <family name> VERSION 4
```

Any other parameters that are currently allowed on the ALTER FAMILY RENAME command can also be specified. The family name does not have to be changed.

To create a version 3 pack or family from a version 4 pack or family, use either the ALTER PACK or the ALTER FAMILY syntax as explained above and specify 3 in the VERSION attribute.

For information on converting shared packs, refer to "Shared Version 4 Packs" in this section.

You can also use the ALTER NEW PACK and the ALTER FAMILY REBUILD commands to convert a Version 3 pack or family to a version 4 pack or family by specifying 4 in the VERSION attribute and using the current syntax for any other attributes that are desired. However, when you use this method new structures are created on the pack resulting in the loss of all file information.

Coexistence of Pack Versions

Different pack versions can coexist under the following guidelines:

- Packs with the same name belong to the same family.
- All packs within a particular family must be the same version.
- Families of version 2 and greater are limited to 15 members.
- Packs in the system resource family must all be the same version.
  - If the system resource family is version 1, multiple resource families are allowed.
  - If the system resource family is version 2 or greater, there can be only one resource family.

Coexistence of MCP Versions and Pack Versions

MCP releases prior to ASR 3.3 support only certain pack versions. For example, since version 2 packs were introduced at ASR 3.1, any release prior to 3.1 does not support version 2 packs. If a pack version is not supported by a particular MCP release, the pack is made not ready and an error message indicating that the label type is invalid is displayed. In order to access existing files on the pack, an MCP release that supports the pack version must be running on the system.

Section 32
OBJCHK—File Compression Utility Program

Overview

The OBJCHK program compresses and decompresses disk or pack files. You can use, at the execution of the utility program, the FILE EQUATE system command to specify the

- Input file to be compressed or decompressed
- Resulting output file

During compression, the OBJCHK program translates files into a combination of ASCII and binary characters so that the resulting compressed files can be transmitted safely across existing data communication systems. OBJCHK decompression performs the reverse translation, restoring the original versions of the files from their compressed forms.

File specification

The OBJCHK program operates on two external file names:

- LARGE (uncompressed files)
- SMALL (compressed files)

When compressing a file, OBJCHK copies and compresses file LARGE to file SMALL. When decompressing a file, OBJCHK copies and decompresses file SMALL to file LARGE.

To specify any file on disk or pack, the FILE EQUATE (FILE) command must be used. The default file type for files LARGE and SMALL is DISK, so the override DPK must be used in file equate statements for files residing on pack. Refer to Volume 2 for more information on this command.

LARGE and SMALL can be the same file, but both file-equate statements for LARGE and SMALL must be specified at the time of execution of OBJCHK.

Operating Instructions

You can execute OBJCHK from an ODT.

Compressing a File

To copy and compress <file1 - id> to <file2 - id>, use the following syntax:

```
EX OBJCHK; VA 0 1; FILE LARGE = <file1-id>; FILE SMALL = <file2-id>
```

The VALUE command sets switch 0 to cause the program to perform compression.

Note: *The OBJCHK program replaces any existing file <file2 - id> with the compressed version of <file1 - id>.*

After the completion of the compression, the ODT displays the following:

```
OBJCHK=<mix#> >> <file1-id> on <multifile1-id> compressed as
                 <file2-id> on <multifile2-id>
OBJCHK=<mix#> >>Compression Ratio = <ratio> to 1
```

Note: *The system dump file ($s0001, where s = system number) cannot be compressed by the program due to MCP restraints. OBJCHK opens all files to be compressed with a lock access to prevent updating of the file during compression. The system dump file, however, cannot be opened with lock access, since the system requires the file to always be available the MCP. To bypass this constraint, you can copy the dump file to a different name with the SYSTEM/COPY utility through the COPY system command, and then compress that file.*

Decompressing a File

To copy and decompress <file2 - id> to <file3 - id>, use the following syntax:

```
EX OBJCHK; FILE SMALL = <file2-id>; FILE LARGE = <file3-id>
```

Note: *The program replaces any existing file <file3 - id> with the decompressed version of <file2 - id>.*

After the completion of the decompression, the ODT displays the following:

```
OBJCHK=<mix#> >> <file2-id> (<file1-id>) on <multifile2-id>
             decompressed as  <file3-id>  on <multifile3-id>
```

In the response, <file1 - id> is the name of the file that originally produced the compressed file, <file2 - id>.

Examples

```
EX OBJCHK; VA 0 1; FILE LARGE = MYFILE; FILE SMALL = MYFILE
```

This compresses the file MYFILE on disk and replaces the original file with the compressed version.

```
EX OBJCHK; VA 0 1; FILE LARGE = WORK/TEST1 DPK; FILE SMALL = TEST2
```

This compresses the file TEST1 on the pack WORK and places the compressed version in the file TEST2 on disk.

```
EX OBJCHK; FILE SMALL = BBB; FILE LARGE = PUBLIC/AAA DPK
```

This decompresses the file BBB on disk and places the uncompressed version in the file AAA on the pack PUBLIC.

Error Messages

Error messages are displayed on the ODT when OBJCHK cannot complete successfully. The error messages are as follows:

```
** Abnormal Termination
```

A premature EOF is encountered during decompression. A program memory dump is produced.

```
**Checksum Error: Record No <record no>
```

An internal decimal-to-binary conversion error occurred on the record numbered <record no>. A program memory dump is produced.

```
**File Structure Error
```

An incorrect decompression of a file occurred. A program memory dump is produced.

```
**Input File Invalid Format
```

Either the file SMALL is not a compressed file, or the file LARGE exceeds a record size of 40,000 digits when compression is executed.

```
**Input File Not Available
```

Either the file equated with LARGE does not exist in the location specified when compression is executed, or the file equated with SMALL does not exist in the location specified when decompression is executed.

Section 33
VFUGEN—Vertical Format Unit File Builder

Overview

The vertical format unit file builder program, VFUGEN, allows you to create, edit, print, and save an electronic vertical forms (EVF) file. The EVF file can be downloaded to a buffered printer that has electronic vertical forms unit (EVFU) capability. The electronic vertical forms unit capability takes the place of mechanical vertical formatting units. Setting up the electronic vertical form is the equivalent of punching channels in the paper tape of a mechanical unit.

VFUGEN is an interactive utility. You use screens that VFUGEN presents to tell VFUGEN what to do.

You can operate VFUGEN in the following ways:

- from the V Series Communication System (VCS)
- from the OCS or ODT
- from versions of V Series CANDE that support initiation of timesharing tasks from the CANDE EDITOR.

The VFUGEN utility must be stored on disk. It can read and write both disk and diskpack files.

Initiating VFUGEN

Initiate VFUGEN from a data communications terminal or from the terminal acting as the OCS or ODT. The VFUGEN screens will display at the same terminal.

Initiating VFUGEN From VCS

You initiate VFUGEN from VCS using a transaction that you name and declare. The following text describes how to declare VFUGEN to VCS, and how to create a transaction for initiating VFUGEN from a VCS station.

If you follow the suggestions here, you initiate VFUGEN with the following VCS transaction:

```
/VFUGEN-27
```

Declaring VFUGEN to VCS

Use the APPL-ADD transaction to declare VFUGEN to VCS. You can choose any name you like for the application; this example uses VFUGEN-27. The APPL-ADD transaction would be the following:

```
/APPL-ADD,VFUGEN-27
```

For more information on the APPL-ADD transaction, consult the *V Series VCS Implementation Reference Manual Volume 2: System Transactions.*

When you enter the APPL-ADD transaction, VCS displays the Applications screen. Figure 33–1 shows the values that should be entered in this screen. Use your own values where asterisks (****) are shown. For details, consult the *V Series VCS Implementation Reference Manual Volume 2: Entities and Screens.*

```
PG.  1   ADD    APPLICATION    26    14:29:11  ON   01/21/94     VCS  NC
FACILITY
ACTION:   [                                                              ]

  USERCODE:  BENSON     VERSION:  VCS27I ON MPV027                  FORM:  609

  NAME  [VFUGEN-27                              ]          PROG-ID [VFUGEN]
  USERCODE  [****       ]    PASSWORD [****       ]            CG [****  ]
  EXECUTION HOST                    [your-host-name****                    ]
  BOJ STATUS (UP,DOWN)          [DOWN]    MODE (FULL/PART) [FULL]  SINGLE USER [Y]
  QUEUE MSG IF APPL IS DOWN       [Y]     UP WHEN THRESHOLD EXCEEDED (Y/N)     [Y]
  QUEUE THRESHOLD TO UP APPL    [  0]     ON IN ONE OUT (Y/N)                  [N]
  ROLL OUT CRITERIA (MINUTES)   [  0]     IDLE DOWN TIMEOUT (MINUTES):      [  5]
  APPLICATION DOWN FORM INDEX   [  0]     DOWN ASSIGNED STN IF APP DOWN (Y/N)  [N]
  ALWAYS UPDATE CONTINUATOR (Y/N)[Y]      SUPPRESS CONTINUATOR (Y/N)           [N]
  APPL TO APPL ALLOWED (Y/N)      [N]     PRIVILEGED APPLICATION (Y/N)         [N]
  HEADER TYPE                 [GEMCOS]    ASSIGN NOTIFY (Y/N)                  [N]
  COMM TYPE (STOQ,CRCR,PORT)   [CRCR]     AC RESPONSE SEPARATE PATH (Y/N)      [N]
  STOQUE OPTNS: QUEUE LIMIT [ 0]  SND CRCR WAKEUP (Y/N) [N]  CLR QUEUE (Y/N) [N]
     MCP CONTROL COMMANDS  [                                                  ]
     (VALUE, INSERT, FILE, [                                                  ]
      PRIORITY, ETC.)      [                                                  ]
```

Figure 33–1. Declaring VFUGEN to VCS

Creating a VCS Transaction for VFUGEN

Use the TRAN-ADD transaction to create a VCS transaction for initiating VFUGEN. You can choose any name you like for the transaction; this example uses VFUGEN-27. The TRAN-ADD transaction would be the following:

```
/TRAN-ADD,VFUGEN-27
```

For more information on the TRAN-ADD transaction, consult the *V Series VCS Implementation Reference Manual Volume 2: System Transactions.*

When you enter the TRAN-ADD transaction, VCS displays the Transaction screen. Figure 33–2 shows the values that should be entered in this screen. Use your own values where

asterisks (****) are shown. For details, consult the *V Series VCS Implementation Reference Manual Volume 1: Entities and Screens.*

```
      ADD    TRAN    104    14:54:40  ON   01/21/94            VCS  NC FACILITY
ACTION:   [                                                                    ]

  USERCODE:  BENSON     VERSION:  VCS27I ON MPV027                    FORM:  50

  TRANSACTION NAME      [VFUGEN-27    ]       SECURITY GROUP NAME         [****]
  ROUTING:  APPLICATION (A), APPLICATION GROUP (AG), COMM.  PROC.  (CP),    [ A]
            APPLICATION CONTROLLER (AC) OR NETWORK CONTROLLER (NC)
  SPECIAL ROUTING (LO = LOGON, LK = LINK, CL = CLEAR)                       [LO]
  APPL/APPL GROUP NAME [VFUGEN-27                                  ]

  BOJ STATUS (UP, DOWN)          [  UP]          DCS TRAN CODE              [  0]
  AUDIT (YES,NO)                 [   N]          I/O MODE (FULL,PART)      [FULL]
  APPL-LINK WELCOME FORM NUMBER  [   0]    TRANSACTION TIMEOUT (SECS)      [999]
  TRANSACTION TYPE                         TEST MODE (YES,NO)             [  N]
        (NDL,ITBR,CHAIN,GEMCOS) [ITBR]     INITIATOR BYPASS FORM INDEX    [   ]
BREAKOUT NOTIFY (YES,NO)          [  N]    CHAIN NUMBER                   [   0]
PRIORITY (IF TYPE = CHAIN)        [  0]    PARSE START OFFSET             [   0]
PARSING LENGTH                    [  0]
HOST FOR ROUTING  [                                    ]
 (VALID ONLY FOR ROUTING TYPE OF AC OR NC)
```

Figure 33–2. Creating a VCS Transaction for VFUGEN

Initiating VFUGEN from CANDE

You can initiate VFUGEN from the CANDE EDITOR if you are using version of CANDE and the CANDE EDITOR that support the SYSTEM command. This version of CANDE is commonly called OLD-CANDE.

From the CANDE EDITOR, use the following command:

```
SYS VFUGEN
```

Note: *VFUGEN cannot be initiated from the CANDE Editor if you are using a version of CANDE (NEW-CANDE) that runs under VCS. If you are using this version of CANDE, initiate VFUGEN directly from VCS.*

Initiating VFUGEN from the ODT or OCS

If you are at the operator display terminal (ODT) or operator control station (OCS), you can initiate VFUGEN with the following command, where *ccu* is the channel number and unit number of the ODT or OCS. Note that there is **no** slash between the channel number and unit number.

```
EX VFUGEN (<ccu>)
```

For example, the following command would initiate VFUGEN on an ODT or OCS that is unit 0 on channel 27:

```
EX VFUGEN (270)
```

Operating VFUGEN

All of the VFUGEN screens have the same basic format.

- At the top of each screen is the following:
  - The title of the screen
  - A field labeled "Action"
  - A list of actions you can put in the "Action" field. The capitalized letters show how you can abbreviate the command.
- The body of the screen has fields for you to enter your choices.
- Error messages are displayed at the bottom of the screen.

The actions that you can put in the Action field are described in Table 33–1. Note that not every action can be used in every screen.

Table 33–1. Actions for VFUGEN Screens

Action	Description
ABort	Terminates the VFUGEN utility immediately and returns you to the system environment. The work done during the VFUGEN session is lost.
CHange	Writes the text on the screen to the output file. Use this action to modify, add, or create a line.
DELete	Deletes a record from the output file. (A record is equivalent to a line.)
End	Terminates the VFUGEN utility and saves your input in the output file. If the medium you select for your output file is not available when you select End, you are given the opportunity to redirect the output to another medium.
HElp	Explains how to fill in the screen.
HOme	Returns you to the Master Selection Menu.
NExt	Displays the next screen with additional information, or a fresh screen.
PREvious	Displays the previous screen. If there was no preceding screen, the current screen is redisplayed.
REFresh	Redisplays the current screen, discarding any data that was changed since the original display.

Welcome Screen

The Welcome screen is the first screen displayed.

Press the Xmit key to display the File Specification Screen.

```
                        Unisys Corporation

                    Vertical Format Unit Builder

Please set your terminal to upper case only, no scrolling.
Either set terminal to full screen transmit or home the cursor before
transmitting.
All data should be in upper case (except filenames where appropriate).

                                                  Hit XMT to continue [ ]
```

Figure 33–3. VFUGEN Welcome Screen

File Specification Screen

The File Specification screen is the second screen displayed. It lets you select the EVF files that you want to work with.

```
         File Specification Screen                          VFUGEN
Action:  [          ]
         HElp  REFresh  ABort

   Input File
         File Id:                    [      ] ON  [      ]
   Output File
         File Id:                    [      ] ON  [      ]

   For disk/pack, use <filename> on <familyname>.
   If output file not specified, input file is default
   output file.
```

Figure 33–4. VFUGEN File Specification Screen

Input File

If you provide the name of an input file, VFUGEN reads that file and displays the records it contains.

If the input file is on disk, enter the file name only. If the input file is on diskpack, enter the diskpack family name in the ON field.

If you are creating a new EVF file from scratch, do not enter an input file name.

Output File

The output file is the EVF file you are creating.

If the output file is to be on disk, enter the file name only. If the input file is to be on diskpack, enter the diskpack family name in the ON field.

If you enter an input file name but do not enter an output file name, the input file name is used for the output file.

If the medium (diskpack family or disk) you select for the output file is not available when you select End, the Redirect Output Screen is displayed so that you can select a different medium.

Redirect Output Screen

If the medium (diskpack family or disk) you select for the output file is not available when you select End, the Redirect Output screen is displayed so that you can select a different medium.

If the output file is to be on disk, enter the file name only. If the input file is to be on diskpack, enter the diskpack family name in the ON field.

You can also use this screen to change the name of the output file. If you leave the File Id field blank, VFUGEN uses the name you put in the File Specification Screen.

```
          Redirect Output Screen                          VFUGEN
Action:  [          ]
         HElp  REFresh  ABort

   Output File
          File Id:                    [      ] ON  [      ]

   For disk/pack, use <filename> on <familyname>
```

Figure 33–5. VFUGEN Redirect Output Screen

Master Selection Menu

The Master Selection screen is the main menu of the VFUGEN utility. It displays after you successfully specify the name and medium of the output file and, if applicable, the input file.

Choice 1 displays the Change Line/Channel Screen so that you can electronically punch a channel for a specific line.

Choice 2 displays the Delete Line Screen so that you delete a line from the output file.

Choice 3 prints the output file to a print file. This printer backup file contains the entire electronic vertical forms (EVF) file. When you end your VFUGEN session, the printer backup file ID is displayed.

To make a choice, type the corresponding number in the Selection Number field and press the Xmit key.

```
         Master Selection Menu (home screen)                  VFUGEN
Action:  [           ]
         HElp  REFresh  End  ABort

   Commands:
   (1)  Change (Add, modify or create)
   (2)  Delete Line
   (3)  Print File
                                    Selection Number:  [ ]
```

Figure 33–6. VFUGEN Master Selection Menu Screen

Change Line/Channel Screen

The Change Line/Channel screen allows you to electronically punch a channel for a specific line.

```
         Change Line/Channel Screen                           VFUGEN
Action:  [CHange    ]
         HElp  HOme  REFresh  ABort  PREvious  NExt  CHange

Total form length   [   ] lines    (max valid value is 144)
                                   e.g.  18 inch form at 8 lpi

 LINE
NUMBER                              CHANNELS

             2    3    4    5    6    7    8    9   10   11   12

[   ]       [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
[   ]       [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
[   ]       [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
[   ]       [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
[   ]       [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
[   ]       [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
[   ]       [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
[   ]       [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
[   ]       [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
[   ]       [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
```

Figure 33–7. VFUGEN Change Line/Channel Screen

If you need more columns or a fresh screen, use the NExt action.

To punch any channels on any lines, do the following:

1. Put a value in the Total form length field. This value is required. The maximum is 144 lines.
2. Enter the line number in the LINE NUMBER field.
3. Enter any character in the input field corresponding to the channel you want to punch.

4. Continue adding line numbers and marking channels until you are finished punching. If you need more lines, type CHange in the Action field and press the Xmit key.
5. When you are finished punching, type CHange in the Action field and press the Xmit key.

If you need to delete a line and its channel punches, use the Delete Line screen. Blanking out the line number and the punches does not delete the line from the output file.

Duplicate line numbers are not allowed.

Zero is not a valid line number.

Each line for which you enter a line number must have at least one channel punched.

The maximum number of lines that can be punched is 62. The highest line number used must not be higher than the Total Form Length specified.

Channel 1 is used exclusively to define Top of Form. Line 1 is reserved for a channel 1 punch only; therefore you cannot punch on line 1. (This is a printer limitation.)

Channels 2 through 11 do not have any special requirements, and may be punched in any combination.

Channel 12 is used for bottom-of-form sensing. One line must have channel 12 punched. This line must be the last line, and can have no other channels punched. The line number of this line must be less than the value used for Total Form Length.

Delete Line Screen

Use the Delete Line screen to delete a line from the output EVF file or to change the total form length.

To delete a line, put any character in the field at the left of the line, and press the Xmit key.

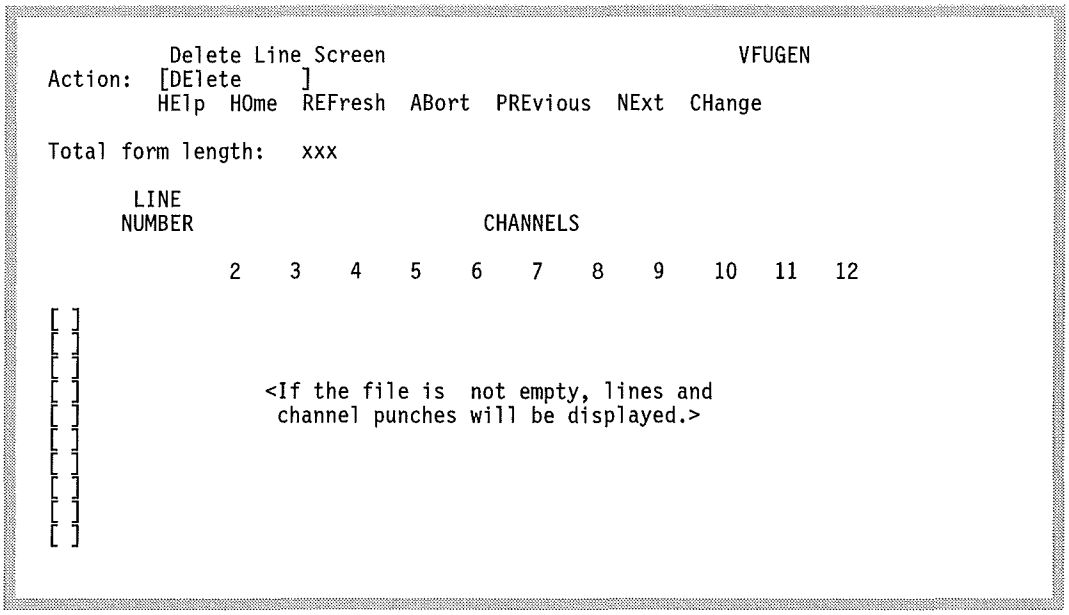


Figure 33–8. VFUGEN Delete Line Screen

Section 34
TRKTAP—TRAK to Tape Program

Overview

The TRKTAP program allows you to collect data that is output by the TRAK diagnostic facility. It frees you from the 999 KD size limit of the TRAK buffer in system memory by transferring the TRAK data to tape.

The TRKTAP program also creates reports based on the collected TRAK data. You can select the TRAK data to be reported on, and the format of the reports printed.

Other than data collection to tape and report generation, the operation and use of TRAK are not affected by the TRKTAP program.

Buffer Management

There are three buffer limits involved with TRKTAP:

- TRKTAP maximum buffer size. This is the maximum size of each of the two buffers that are used to collect TRAK data for transfer to the TRKTAP program. You can set the TRKTAP maximum buffer size when you initiate TRKTAP. The range is 20 KD through 466 KD. The default is 100 KD.
- TRKTAP buffer trigger point. The TRAK facility puts data into the TRKTAP buffer. When the amount of data in the current TRKTAP buffer reaches the trigger point, the MCP transfers the current buffer to the TRKTAP program. The proper adjustment of the trigger point will ensure that the MCP always transfers the buffer before the buffer overflows.

  You can set the TRKTAP buffer trigger point when you initiate TRKTAP. The default size is 50 KD.

- LIMIT TRAKBUFFER. The TRAKBUFFER LIMIT is the size of the TRAK buffer in system memory. It is set by the LIMIT TRAKBUFFER record in the system configuration file and by the ALTER TRAKBUFFER system command. The maximum is 999 KD. The default is 20 KD. The setting can be displayed with the SHOW TRAK system command.

It is recommended that the LIMIT TRAKBUFFER value be at least twice the TRKTAP buffer size.

The TRAK buffer in system memory is a circular buffer, and when the buffer becomes filled, new data begins overwriting existing data. The intent of the TRKTAP program is intended to prevent this loss of data. If newer TRAK data overwrites earlier TRAK data

before the data can be transferred to tape, a missing data warning message is displayed. Reports can still be generated if there is missing data, but if three overwrites occur in succession, then the TRKTAP utility waits for an AX command before continuing.

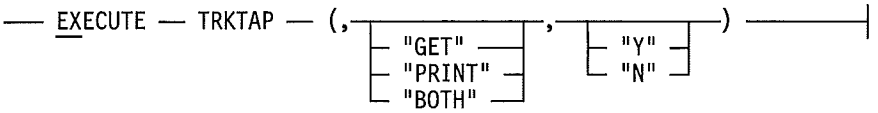
If the missing data warning message occurs, investigate the following possibilities:

- The TRKTAP program might be running at too low a priority on a processor-bound machine. It is recommended that the TRKTAP program execute at priority 9.
- The TRAK buffer in memory is not large enough, and should be increased by increasing the LIMIT TRAKBUFFER value.
- The TRKTAP buffer trigger point and the TRKTAP maximum buffer size are too small. They should be adjusted to accommodate the tape device used.
- TRKTAP expects a scratch tape to be mounted and ready when it switches reels. If the TRKTAP buffer fills before reel switching can be done, data can be lost. Be sure that a scratch tape is available at reel switch time.

Operating Instructions

1. If appropriate, use the ALTER TRAKBUFFER command to change the size of the TRAK buffer in system memory. Note that the TRAK option must be reset to use the ALTER TRAKBUFFER command. You can use the TO TRAK command to check the status of the TRAK option, and the RO TRAK command to reset the TRAK option.
2. Be sure the TRAK option is set. You can use the TO TRAK command to display the status of the TRAK option. If the TRAK option is not set, use the SO TRAK command to set it. If the TRAK option is not set when the TRKTAP program initiates, then the TRKTAP program waits.

   You can use the BT command to initiate a TRAK session before you initiate the TRKTAP program. When you initiate the TRKTAP program, the TRAK data will be directed to tape.
3. Initiate the TRKTAP program, using the EXECUTE command with parameters. The syntax is as follows:



"GET"	This selects the data collection function. This is the default. Note that the leading comma is required.
"PRINT"	This selects the report preparation function. TRKTAP will display options for selection of TRAK data and formatting of reports.
"BOTH"	This selects both the data collection and report preparation functions.
"Y"	This selects the default sizes for the TRKTAP buffer size and the TRKTAP buffer trigger point. This is the default.

"N" This requests special settings for the TRKTAP buffer size and the TRKTAP buffer trigger point. The TRKTAP program will prompt you for the values you want to use.

4. Use the BT (Begin TRAK) command to select a set of TRAK calls and cause the TRAK facility to begin tracking those calls. (If a TRAK session is already active, this step is optional.)
5. Use the ET (End TRAK) command when you are ready to terminate the TRAK session. TRKTAP will receive all the TRAK data and then will be notified that the TRAK session has terminated. At this point TRKTAP closes its tape file and terminates.

   You can terminate TRKTAP without terminating the TRAK session by using the following command:

   <mix number of TRKTAP> QT
6. If you have terminated the TRAK session, you can use the RO TRAK command to return to the system the memory used by the TRAK module.

Examples

```
EX TRKTAP (,"GET","Y")
```

This is equivalent to EX TRKTAP. The result is that TRKTAP will open an output tape and wait for TRAK data.

```
EX TRKTAP (,"BOTH")
```

TRKTAP will open an output tape and wait for TRAK data. When TRAK data collection is terminated, TRKTAP will re-open the tape as an input file and prepare reports.

Printing with TRKTAP

After you run TRKTAP and create a tape file containing collected TRAK entries, you can use TRKTAP to format and print a report based on the collected data. If you have created multiple tape files, the report can begin with any of these tapes. There are options for selecting and formatting the TRAK entries.

Instructions for Operating the TRKTAP Printing Functions

Follow this procedure to create reports from collected TRKTAP data:

1. To print in TRKTAP, use the EXECUTE command to initiate TRKTAP with the parameter PRINT:

   ```
   EX TRKTAP (,"PRINT")
   ```

   The TRKTAP program starts, opens a tape file, and prompts for further input:

   ```
   "ENTER OPTIONS, HELP OR END.  "
   ```

2. You use the AX command to enter the options you want. You can make one or more selections in a single AX command. You can select a total of one start time, one stop time, one date, 14 operating system modules, 20 TRAK codes, one I/O channel, and one task number.
3. When you are finished selecting options, use the AX command to enter the word END, as follows:

   ```
   <mix number>AX "END"
   ```

   This terminates your input and starts TRKTAP formatting and printing your data.

   You can also include the word END with other options, as shown:

   ```
   <mix number>AX "ALLF END"
   ```

   In this case TRKTAP begins formatting and printing as soon as you have answered all of the prompts.

Selecting TRKTAP Printing Options

Use the following command to see a list of options:

```
<mix number>AX "HELP"
```

The HELP display is shown as follows.

```
"ALL[F]     - SET ALL PRINT OPTIONS [USE FORMAT HEADINGS]"
"CLEAR      - CLEAR OPTIONS"
"HELP       - THIS DISPLAY"
"END        - END OPTION INPUT"
" "
"HEAD[NF]   - PRINT AND FORMAT HEADER [NO HEADING FORMAT]"
"Q[1/2]     - PRINT AND FORMAT Q1 AND Q2 INFO"
"DATA       - PRINT USER DATA - NO FORMAT HEADINGS"
"STACK[B/P] - PRINT STACK BACK AND STACK PARMS"
"IOM[D/Q]   - PRINT AND FORMAT DEVICE TBL AND SYS Q'S"
" "
"TIME       - SELECT TIME"
"DATE       - SELECT DATE"
"MODULE     - SELECT MODULE"
"CODE       - SELECT CODE"
"CHAN       - SELECT SYS Q ELEMENT CHANNEL"
"TASK       - SELECT TASK"
```

The words in the left-hand column are the commands you use to get the results shown in the right-hand column. The portions in brackets are optional. For example, to print and format the header, if TRKTAP is executing as mix number 22, enter the following:

```
22 AX "HEAD"
```

To print the header without formatting, enter the following:

```
22 AX "HEADNF"
```

The TRKTAP options are discussed below, in groups as they appear on the HELP listing. More discussion is included under the BT command in the *V Series Systems Software Operations Guide Volume 2: System Commands.*

TRKTAP Print Options—First Group

The first group of options affects the operation of the TRKTAP printing function.

Table 34–1. TRKTAP Print Options—First Group

Option	Description
ALL[F]	ALL and ALLF print all available reports, and are equivalent to selecting all of the options in the second grouping. ALLF prints formatted data similar to a DMPANL listing, with format headings that give the field acronyms for the TRAK data. ALL prints unformatted data without the headings. You must either select ALL or ALLF, or else select one or more of the options in the second grouping.
CLEAR	This clears any options that you have already set.
HELP	This repeats the display of options.
END	This must be entered when you are finished selecting options.

TRKTAP Print Options—Second Group

The second group of options selects portions of the TRAK entries to be printed. You must either select one of these options or else select ALL or ALLF.

Table 34–2. TRKTAP Print Options—Second Group

Option	Description
HEAD[NF]	Each TRAK entry has a TRAK header. HEAD prints these TRAK headers. HEADNF prints the headers without formatting.
Q[1/2]	Q1 prints the input/output queue elements to be initiated. Q2 prints the I/O queue elements that have completed. Q prints both kinds of I/O queue elements.
DATA	This prints user data. User data includes various kind of data collected for the different TRAK entries. This data is printed without format headings.
STACK[B/P]	STACKB prints stack traceback data. STACKP prints stack parameters. STACK prints both kinds of stack data.
IOM[D/Q]	IOMD prints the device table. IOMQ prints the system queues. IOM prints both types of Input/Output module data. The output is similar to the output from DMPANL. Selecting DATA or ALL outputs the same I/O module user data without headings, but also outputs user data from other modules.

TRKTAP Print Options—Third Group

The options in the third group select the TRAK entries to be printed. For each of these options that you select, an additional menu displays, prompting you for further input.

Table 34–3. TRKTAP Print Options—Third Group

Option	Description

TIME	This selects TRAK entries by time. It produces the following prompts:
	`"ENTER START TIME (LENGTH 11=MICRO, 8=MILL, 5=SEC) OR 0"`
	`"ENTER STOP TIME (LENGTH 11=MICRO, 8=MILL, 5=SEC) OR H"`
	Each line displays and the TRKTAP program waits for your input. Use AX commands to enter the START time and STOP time as microseconds past midnight (11 digits), milliseconds past midnight (8 digits) or seconds past midnight (5 digits). Enter 0 (zero) for the START time to start from the beginning of the file. Enter H for the STOP time to end with the end of the file.

DATE	This selects TRAK entries by date. It produces the following prompt: `"ENTER DATE (YYYYMMDD)"` Use an AX command to enter the date. For example, January 30, 1995 would be 19950130.

MODULE	This selects TRAK entries contained within one or more operating system modules. It produces the following prompt:
	`"ENTER MODULES, CLR (TO CLR TBL), OR END - VALID MODS: "`
	`"IOM MES DCP VCS NCP PRT NWK JMR DCU RJE STQ MBX CWT ISC SCA "`
	Use AX commands to select up to a maximum of 14 modules.
	The module names are listed on the second line of the prompt. For explanations, refer to the BT command in the *V Series Systems Operations Guide Volume 2: System Commands.*

CODE	This selects TRAK entries by four-digit code. It produces the following prompt: `"ENTER CODES, CLR (TO CLR TBL), OR END"` Use AX commands to enter up to 20 TRAK codes. The TRAK codes and interpretations are printed out by DMPANL in its analysis of the TRAK buffer.
CHAN	This selects I/O queue elements associated with a particular channel. It produces the following prompt: `"ENTER SYS Q CHANNEL (NN) OR ALL"` Use an AX commands to enter a two-digit channel number, or ALL for all channels.

continued

Table 34–3. TRKTAP Print Options—Third Group (cont.)

Option	Description
TASK	This selects TRAK entries that relate to a particular task. It produces the following prompt: `"ENTER TASK (NNNN)"` Use an AX command to enter a four-digit task number.

Example 1

```
EX TRKTAP (,"PRINT");AX"ALLF END"
```

This prints all entries, using formatting similar to DMPANL

Example 2

```
EX TRKTAP (,"PRINT");AX"HEADNF DATA CODE TIME CHAN END"
```

This prints headings without a format line, and prints user data for the selected TRAK entries. You select the TRAK entries by using the prompts for CODE, TIME, and CHAN. These prompts, and example responses, are shown below, in the order in which TRKTAP presents them.

Prompt

```
"ENTER SYS Q CHANNEL (NN) OR ALL"
```

Sample Response

```
<mix number>AX"04"
```

Result

System queue user data is printed only if it is from channel 4.

Prompt

```
"ENTER START TIME (LENGTH 11=MICRO, 8=MILL, 5=SEC) OR 0"
```

Sample Response

```
<mix number>AX"28800"
```

Result

TRAK entries after 8:00 a.m. are selected for printing.

Prompt

```
"ENTER STOP TIME (LENGTH 11=MICRO, 8=MILL, 5=SEC) OR H"
```

Sample Response

```
<mix number>AS"28860"
```

Result

TRAK entries before 8:01 a.m. are selected for printing.

Prompt

```
"ENTER CODES, CLR (TO CLEAR TBL), OR END"
```

Sample Response

```
<mix number>AX"9910 END"
```

Result

TRAK code 9910, "I/O Complete (PIQed)," is selected for printing.

The result of this example is that TRKTAP prints user data and headers for I/O completes on channel 04 from 8:00 a.m. to 8:01 a.m.

Example 3

```
EX TRKTAP (,"PRINT");AX"HEAD MOD END"
```

This prints headings with a format line for the selected TRAK entries. You select the TRAK entries by responding to the prompt for MOD. The prompts and a sample response follow:

Prompts

```
"ENTER MODULES, CLR (TO CLEAR TBL), OR END - VALID MODS: "
IOM MES DCP VCS NCP PRT NWK JMR DCU RJE STQ MBX CWT ISC SCA "
```

Sample Response

```
<mix number>AX"PRT END"
```

Result

This selects TRAK entries from the PORT module of the operating system. All TRAK entries from the PORT module are selected for printing.

Selecting Options with Switch Settings

You can use programmatic switches to set TRKTAP printing options without entering AX commands. The values are bit coded so that a single switch can be used for multiple options. You will still be prompted for relevant values.

Table 34–4. TRKTAP Programmatic Switches for Printing Options

Digit	Meaning or Values
0	Change options after printing has begun.
1	1 = Print header 2 = Print user data
2	1 = Print stack parameters 2 = Print stack traceback data
3	1 = Print I/O operations to be initiated (Q1) 2 = Print I/O operations completed (Q2)
4	Formatting options: 1 = Print headings with the system queues 2 = Print headings with the device table 4 = Print headings with the TRAK headers 8 = Print headings with the Input/Output queues (Q1/Q2)
5	1 = Prompt for selection by time 2 = Prompt for selection by date
6	1 = Prompt for TRAK code 2 = Prompt for operating system module 4 = Prompt for I/O channel 8 = Prompt for task number
7	(Reserved for debugging use)

Examples

The following two examples produce the same results:

```
EX TRKTAP (01004020,"PRINT")

EX TRKTAP (,"PRINT");AX"HEAD MOD END"
```

The following two examples produce the same results:

```
EX TRKTAP (03000150,"PRINT")

EX TRKTAP (,"PRINT");AX"HEADNF DATA CODE TIME CHAN END")
```

Section 35
TAPDIR—Tape Directory Report Program

Overview

The TAPDIR program analyzes tapes produced by the following utilities:

- SYSTEM/COPY
- LOADMP
- PACKUP

The analysis includes a list of the files on each tape. The analysis is provided as a printer backup file.

Operating Instructions

Initiate the TAPDIR utility using the EXECUTE system command. The name of the tape to analyze is the first slash parameter, as follows:

```
EXECUTE TAPDIR/<name of tape>
```

Alternatively, initiate TAPDIR without a slash parameter. The system will request a tape. Use the IL system command to provide the channel and unit number of the tape drive. The IL system command is described in the *V Series Systems Operations Guide Volume 2: System Commands.*

Output

The TAPDIR program produces a printer backup file with the internal file name FILE and the external file name TAPDIR. The listing includes the following information:

- For the tape:
  - Date of TAPDIR execution
  - The name of the program that produced the tape (SYSTEM/COPY, LOADMP, PACKUP)
  - The name, serial number, reel number, and density of the tape analyzed, and the channel number and unit number on which it was mounted while TAPDIR was analyzing it

  - The date on which the tape was created, and its purge date

- For each file on the tape:
  - The name of the medium on which the file originated (for example, the disk or diskpack family name), and the type of medium on which it originated (for example, DISK or PACK).
  - The title (name) of the file. This is the value of the FILENAME file attribute.
  - The date and time at which the file was created.
  - The type of file (COBOL, DATA, etc.). The letter [C] indicates that the file was created by the CANDE Editor.
  - The record size, in bytes.
  - The number of records per block.
  - The maximum number of areas allowed for the file. This is the setting of the AREAS file attribute.
  - The area size. This is the value of the AREALENGTH file attribute.
  - The number of records in the file.
  - The amount of memory required for executable program files.
  - The usercode under which the file was created.
  - The setting of the SECURITYTYPE file attribute.
  - The setting of the SECURITYUSE file attribute.

If SYSTEM/COPY created the tape and was given a command to transfer a file to the tape, but TAPDIR does not find the file on the tape, the TAPDIR listing will show the following message:

```
FILE SKIPPED OR NOT PRESENT AT TIME OF TRANSFER
```

The following circumstances can produce this condition:

- The SYSTEM/COPY syntax included an EXCEPT phrase.
- An attempt was made to copy a mirrored user file (USERFL).
- The file was removed after the COPY command was entered but before the file was physically copied.

File attributes are discussed in the *V Series MCP/VS Programming Reference Manual.* The SECURITYTYPE and SECURITYUSE file attributes and the use of usercodes are discussed in the *V Series System Security Installation and Operations Reference Manual.*

Examples

The following example initiates TAPDIR to analyze the directory of the tape named MYTAPE.

```
EX TAPDIR/MYTAPE
```

The following example shows the use of the IL command. If you do not specify a tape when you initiate TAPDIR, TAPDIR opens an input tape with the internal file name TAPEIN and the external file name TAPEIN/. You can then use the IL command to direct TAPDIR to the desired tape name or tape drive.

In this example, TAPDIR enters the schedule under mix number 027. It begins executing at 8:34 a.m. TAPDIR waits on a NO FILE condition, waiting for the file TAPEIN/. The IL command directs TAPDIR to the tape that is mounted on the tape drive at channel 6, unit 1.

```
EX TAPDIR

==> EX TAPDIR
     TAPDIR=027 SCHED
     BOJ TAPDIR=027 032394 08:34
     ** NO FILE TAPEIN/ MTP RL# 001 TAPDIR=027

27 IL 6/1

==> 27 IL 6/1
```

Section 36
DPKANL—Diskpack Analysis Program

Overview

The DPKANL program has evolved as the primary diagnostic tool used by Unisys personnel for validating and investigating problems with diskpack structures.

Cautions

1. The DPKANL program is provided solely as a diagnostic aid. It should be used only in extreme cases under the supervision of Unisys support personnel. Great caution should be exercised in its use.
2. This program has the ability to patch diskpack sectors and therefore may pose a risk to the integrity of your data and of the diskpack structures.
3. This program has the ability to display data directly from diskpack sectors and therefore may pose a security risk.

Initiating DPKANL

Initiate DPKANL with the EXECUTE command. Include an INSERT command that encodes the functions desired, as listed in Table 36–1.

If you do not include the INSERT command with the EXECUTE command, DPKANL displays a list of its functions and an ACCEPT message. Perform a *<mix number> IN* command to provide the information and a *<mix number> AX* command to proceed, as shown in the second example at the end of this section.

An INSERT command includes a data string known as the INSERT string. The INSERT string for DPKANL is a five-digit string starting at address 0. It can contain digits (0–9) and/or undigits (A–F). You construct the insert string from the bit meanings given in Table 36–1.

For example, to request that DPKANL print the master available table, you would enter the following:

```
EXECUTE DPKANL; INSERT 0 5 UN 40200
```

When DPKANL starts, it requests the channel number/unit number of the diskpack you want to work with. Enter this information with an AX command, using the format *<channel number>/<unit number>*, including the slash.

The first action DPKANL takes is to perform an OL command to ensure that the device is a diskpack. Then the diskpack is saved, because Direct I/O is used for all reads and writes. The pack remains saved when DPKANL completes.

In general, DPKANL detects and displays read and write errors.

Table 36–1 gives the meaning of each bit in each digit of the INSERT string.

Table 36–1. DPKANL INSERT String Bit Values

Digit:Bit	Meaning
0:8	Format and print the volume label.
0:4	Format and print the master available table.
0:2	Format and print the available table.
0:1	Not used.
1:8	Format and print the master available table and the available table.
1:4	Format and print the full diskpack directory.

1:2	Format and print a particular file header.
	When you request this function, DPKANL displays the following lines, and waits for your AX response after each line:
	`ENTER DIRECTORY ADDRESS - 8 DIGITS`
	`ENTER DIRECTORY ENTRY NUMBER - 4 DIGITS`
	`ENTER FILE HEADER ADDRESS - 8 DIGITS`
	When it receives all of your input, DPKANL reads the requested directory sector. It then outputs some information from the directory, and then reads, formats, and prints the file header.

1:1	Print all file headers. For this function, DPKANL goes through the directory, following each file entry to its header. Each file header is then formatted and printed.
2:8	Print all volume structures. This function formats and prints the following structures: • Label • Directory • Master available table • Available table

continued

Table 36–1. DPKANL INSERT String Bit Values (cont.)

Digit:Bit	Meaning
2:4	Obtain a structure address through an ACCEPT. This function can be used in conjunction with any other function that requests a structure analysis. It allows you to input the structure address in situations where DPKANL cannot obtain the structure address from the diskpack itself. DPKANL issues an ACCEPT message, and then you enter the structure address with an AX command. For example, you might want to analyze the directory on a diskpack that has a smashed label. Since DPKANL cannot obtain the directory address from the diskpack label, you would include this request (2:4) along with the directory analysis request (1:4). DPKANL requests the directory address and then proceeds with the analysis.
2:2	**Always set this bit.** It indicates that the INSERT string is valid. If you do not set this bit, DPKANL ignores the INSERT string and again displays the list of functions and the ACCEPT message.

2:1	Display and patch sectors.
	
	This is a basic implementation of the standard DISPLAY ABSOLUTE and PATCH ABSOLUTE commands as applied to diskpacks. It allows you to display and patch diskpack data by sector number and offset within the sector.
	
	When you request this function, you see the following ACCEPT display:
	
	`DISPLAY <SECTOR> <OFFSET> <LENGTH>`
	`PATCH  <SECTOR> <OFFSET> <LENGTH> <DATA>`
	`AX END TO QUIT`
	
	Respond by entering the word DISPLAY or PATCH followed by the relevant data. This function loops until you enter the word END, so that you can do multiple displays or patches as needed.
	
	The <sector> is zero-relative. The <offset> and <length> are in digits. The <data> is UN (unsigned numeric digits).

continued

Table 36–1. DPKANL INSERT String Bit Values (cont.)

Digit:Bit	Meaning

3:8	Search for headers.
	
	This function searches for sectors that resemble header structures (structures in which header self-pointer = this address). When one is found, DPKANL formats and prints it, and then continues the search. DPKANL keeps a count of the total number found, displaying the count at the end of the analysis.
	
	When you request this function, DPKANL displays the following lines, and waits for your AX response after each line:
	
	`ENTER START ADDR OR END`
	`ENTER LENGTH OF SEARCH`
	
	In response to the first line, enter the address (sector number) at which to begin the search. In response to the second line, enter the number of sectors to search. To abort the function, enter END in response to the first line.
	
	This function can be used in a situation where directory blocks are corrupted. Stranded headers can be located, and from that information, the directory sectors can be rebuilt manually.

3:4	Print a file header. When you request this function, DPKANL displays the following line and waits for your AX response: `ENTER FILE ID` DPKANL attempts to locate the file identifier in the directory and to obtain the file header address from the directory entry. It then formats and prints the file header.

3:2	Search for directories.
	
	This function searches for groups of sectors that resemble directory structures (directory self-pointer = this address and directory-validity-flag = F). When such a group is found, DPKANL formats and prints the first five sectors, and then continues the search. It keeps a count of the total number found, displaying the count at the end of the analysis.
	
	When you request this function, DPKANL displays the following lines, and waits for your AX response after each line:
	
	`ENTER START ADDR OR END`
	`ENTER LENGTH OF SEARCH`
	
	In response to the first line, enter the address (sector number) at which to begin the search. In response to the second line, enter the number of sectors to search. To abort the function, enter END in response to the first line.
	
	This function can be used in a situation where directory linkages have been accidentally severed. Stranded entries can be located and relinked manually.

continued

Table 36–1. DPKANL INSERT String Bit Values (cont.)

Digit:Bit	Meaning

3:1	Search for partial directories.
	
	This function searches for individual sectors that resemble directory structures (directory self-pointer = this address and directory-validity-flag = F). When one is found, DPKANL displays its address, and then continues the search. DPKANL keeps a count of the total number found, displaying the count at the end of the analysis.
	
	When you request this function, DPKANL displays the following lines, and waits for your AX response after each line:
	
	`ENTER START ADDR OR END`
	`ENTER LENGTH OF SEARCH`
	
	In response to the first line, enter the address (sector number) at which to begin the search. In response to the second line, enter the number of sectors to search. To abort the function, enter END in response to the first line.
	
	This function differs from the previous function in that it can find directory fragments instead of assuming some coherency (five sectors) to the directory.

4:8	Sector copy. This function copies sectors from the source address to the destination address. When you request this function, DPKANL displays the following message and waits for your AX response: `COPY <SOURCE ADDR> <# SECTORS> <DESTINATION ADDR>` In response, enter the word COPY followed by the necessary data. The <source addr> is the zero-relative sector at which to start copying. The <destination addr> is the zero-relative sector at which to start overwriting data. The <# sectors> is the length of data to copy, in sectors. Copying is done 5 sectors at a time (or less on the last I/O) until the request is fulfilled.
4:4	Search for bad headers. This function goes through the directory, following each file entry to its header, checking for the following: • A creation date that contains undigits • A header self-pointer not equal to the directory header address

Example

The following example shows two ways of requesting an analysis of the volume structures (label, directory, and available tables) of a diskpack on channel 54, unit 1.

In the first example, the INSERT command is included with the EXECUTE command.

```
EXECUTE DPKANL; INSERT 5 5 UN 00A00
     DPKANL=029 SCHED
     BOJ DPKANL=029 081694 17:07
     DPKANL=029 ENTER CC/UU OF DEVICE TO ANALYZE.
       ** DPKANL=029 ACCEPT
==> 29 AX "54/01"
==> SV 54/01       (Note that this SV was done by DPKANL.)
     54/1 DPK ID 151 SAVED
     @10613 DPKANL OPEN OUT DPKANL=029
     @10613 DPKANL 20/1 DPK ID 121 LOCKED DPKANL=029
     EOJ DPKANL=029 17:08 IN 00:46 CHG 00:01
```

The second example makes the same request, but uses the prompts from DPKANL.

```
==> EX DPKANL
     DPKANL=029 SCHED
     BOJ DPKANL=029 081694 17:07
     FIRST DIGIT
     BIT 8: PRINT THE VOLUME LABEL
         4:  PRINT THE MASTER AVAILABLE TABLE.
         2:   PRINT THE AVAILABLE TABLE.
         1:    PRINT THE WORKING AVAILABLE TABLE.
     SECOND DIGIT.
     BIT 8: PRINT ALL AVAILABLE TABLES.
         4:  PRINT THE FULL DIRECTORY.
         2:   PRINT A FILE HEADER GIVEN ADDRESS
         1:    PRINT ALL FILE HEADERS.
     THIRD DIGIT
     BIT 8: PRINT ALL VOLUME STRUCTURES
         4:  OBTAIN STRUCTURE ADDRESSES VIA ACCEPT
         2:   VALUE STATEMENT PROVIDED.
         1:    FIX A SECTOR BY PATCHING
     FOURTH DIGIT
     BIT 8: LOOK_FOR_HEADERS
         4:  PRINT A HEADER GIVEN FILE ID
         2:   LOOK FOR DIRECTORIES
         1:    LOOK FOR PARTIAL DIRECTORIES
     FIFTH DIGIT
     BIT 8: SECTOR COPY
     BIT 4: LOOK FOR BAD HEADERS
     'IN' REQUEST TO BASE AND 'AX' TO CONTINUE
       ** DPKANL=029 ACCEPT
==> 29 IN 0 5 00A00; 29 AX
     DPKANL=029 0= 00A00
     DPKANL=029 ENTER CC/UU OF DEVICE TO ANALYZE.
       ** DPKANL=029 ACCEPT
==> 29 AX "54/01"
==> SV 54       (Note that this SV was done by DPKANL, not by the operator.)
     54/1 DPK ID 151 SAVED
     @10614 DPKANL OPEN OUT DPKANL=029
     @10614 DPKANL 20/1 DPK ID 121 LOCKED DPKANL=029
     EOJ DPKANL=029 17:08 IN 00:46 CHG 00:01
```

Section 37
SQUASH—Diskpack Squash Program

Overview

The SQUASH utility can reclaim space and improve performance on a diskpack by consolidating areas and reducing "checkerboarding." This consolidation is referred to as "squashing" the diskpack. SQUASH squashes a diskpack by moving the data and updating the table links, diskpack directory, and file headers.

The SQUASH utility can also reconstruct some of the diskpack tables in the event of failure. The information in the "master available table" and the diskpack directory is used to reconstruct the working available table. Because the directory is the complement of the available table, this rebuilding of tables is called "complementing" the diskpack.

V Series diskpack concepts are discussed in "Pack Subsystems" in the *V Series Systems Operations Guide Volume 3: System Utilities.*

Recovery Tape

You can request that the SQUASH utility create a recovery tape that contains the contents of the diskpack before carrying out a squash. This recovery tape can serve as a backup tape for the diskpack.

If the squashing process fails, run SQUASH again by entering the SQP command again. SQUASH will read the recovery tape and, if possible, will complete the squash and recover the diskpack.

You can create a recovery tape even if you do not need to squash the diskpack. The recovery tape can be used as a backup of the data on the diskpack. Use the VALUE 0 2 syntax in the SQP command to create a recovery tape and compare it to the data on the pack. Use the VALUE 0 3 syntax to create the recovery tape and skip the comparison.

The tape identifier of the recovery tape is constructed from the name and serial number of the diskpack. The tape identifier will be:

```
<diskpack name>/<diskpack serial number>
```

Procedures before Executing SQUASH

The following steps will help to ensure that the operation of SQUASH is safe and effective.

Determine which Diskpacks Should Be Squashed

To determine if a diskpack is a candidate for squashing, use the following command (see the *V Series Systems Operations Guide Volume 2: System Commands* for details):

```
SHOW PACK <cc/u> A <n>
```

This shows in tabular form the amount of checkerboarding present on a diskpack.

If you never have programs waiting for space on a diskpack, then the diskpack might not need to be squashed. Even so, squashing the diskpack might speed up data access, since the data will be located physically closer together, resulting in decreased seek time.

Check the Reliability of the Diskpacks

Before you run the SQUASH utility on a diskpack or diskpack family, you should examine the Diskpack Exception Report and the Diskpack Volume Exception Condition Report to determine if the diskpack or diskpack family is encountering many errors. A diskpack that is encountering an abnormal number of errors during normal processing has a higher probability of encountering an error during a squash. In this case a squash of the diskpack is **not** recommended. Refer to the *V Series System Software Logging Operations Reference Manual* for information on the diskpack exception reports, the Maintenance Log, and the MLGOUT utility program.

Make Backups

Before you squash a diskpack, first back up the diskpack family. Then squash those diskpacks in the family that will benefit from a squash.

Back up the family using whatever method you prefer, including:

- The Disk Pack Copy Utility Program, PKCOPY, discussed in this manual
- The File Transfer Utility Program, SYSTEM/COPY discussed in this manual
- Making a SQUASH recovery tape for each diskpack in the family, using the procedures discussed in this section.

Initiating SQUASH

The SQUASH utility is invoked by the SQP system command. The complete syntax of the SQP command is described in the *V Series Systems Operations Guide Volume 2: System Commands.*

The diskpack must not be in use when SQUASH is initiated. If any files are open on the designated diskpack when SQP is entered, the MCP will display the following message:

```
** KBD IGNORED: UNIT IN USE
```

No other programs can access the diskpack until SQUASH terminates.

Allocating SQUASH Memory

The speed at which data is moved from its old location to its new location is very much dependent on the amount of memory SQUASH is allowed to use. If no extra memory is supplied, then SQUASH uses its own internal buffer, which can contain a maximum of 30 diskpack sectors. Including a MEM 999 clause in the SQP command assigns 999KD of memory to SQUASH, providing a buffer of approximately 2580 diskpack sectors in length. This can dramatically decrease the number of I/Os required to move the data on a diskpack.

Keep in mind that not all I/Os are done at the maximum size. If a MEM 999 clause is used, the I/O size used is the minimum of 2580 sectors or that which is necessary to move a structure, whether it is a directory block (10 sectors), a file header (1–3 sectors) or a data area (1–9999999 sectors).

Access to Diskpacks

You should be aware that executing SQUASH limits the access that other programs have to a diskpack.

Access to the Base Diskpack when Squashing a Continuation Diskpack

For a diskpack family of version 2 or greater, the base diskpack contains all of the directory blocks and headers for all of the files on all of the diskpacks of the family. The continuation diskpacks do not contain any directories or file headers, just data areas. When the SQUASH utility is squashing a continuation diskpack, it needs to know what data is on the diskpack and where it resides. To obtain that information, the SQUASH utility needs exclusive access to the base diskpack of the family. For this reason, when a continuation diskpack is squashed, SQUASH uses both the continuation diskpack and the base diskpack, simultaneously. Therefore only one continuation diskpack in a family may be squashed at a time.

Squashing a Shared Diskpack

Before you run SQUASH on a shared diskpack, reserve the diskpack on all systems except the one on which you will execute SQUASH. A diskpack can be reserved by using the UR+ system command.

When you execute SQUASH on a shared diskpack, SQUASH displays the following message:

```
THE AFFECTED PACK <cc/u> IS SHARED
PLEASE RESERVE ALL PACKS IN FAM ON ALL OTHER SYSTEMS
DO YOU WANT TO CONTINUE (ENTER YES OR NO)?
```

Check to see that the diskpack has been reserved as needed. If you are ready to proceed, enter YES as follows:

```
<mix number of SQUASH> AX YES
```

If you enter *<mix number of SQUASH> AX NO*, then SQUASH terminates.

Refer to the *V Series Systems Operations Guide Volume 2: System Commands* for details of the UR (Inhibit or Uninhibit Unit) command. Refer to the *V Series Systems Operations Guide Volume 1: System Initialization* for information on the RESERVED option of the PACK record of the system configuration file.

Pre-allocating Directory Blocks for Safety

The most dangerous structure to move on a diskpack is a directory block. Each directory block has a backward link to the block preceding it and a forward link to the block following it. This means that three directory blocks actually need to be updated for each block that is moved. The possibility of a diskpack being corrupted accidentally due to I/O errors is greatly reduced if SQUASH does not have to move any directory blocks.

To ensure that future executions of SQUASH do not have to move directory blocks, allocate enough contiguous directory blocks to hold the information for the maximum number of files that you expect to reside on the diskpack family.

To allocate contiguous diskpack directory blocks, use the appropriate FILES syntax in the ALTER NEW PACK or ALTER FAMILY commands for diskpacks of version 2 or greater, or use the FILES syntax in the DISPKV command string for version 1 diskpacks.

Note that when you use the FILES option of the ALTER NEW PACK command, the ALTER FAMILY command, or the DISPKV utility, all files on the diskpack are purged.

Using SQUASH to Recover from Diskpack Error Conditions

Some diskpack problems can be fixed without loss of data by using the SQUASH utility and a recovery tape. These conditions are discussed under "Rebuilding Diskpack Structures" and "Recovering to a Different Spindle" below. More serious problems require using the ALTER command or the diskpack utility program, DISPKV. The DISPKV utility is discussed in this guide. The ALTER command is discussed in the *V Series Systems Operations Guide Volume 2: System Commands.*

Rebuilding Diskpack Structures

By rebuilding the "working available table," the SQUASH utility can perform a certain amount of recovery on a troubled diskpack. You can have SQUASH rebuild the "working available table" without squashing the diskpack by using the SQP command with the syntax *VALUE 0 C* or the syntax *VALUE 0 F.*

If the diskpack label is unreadable, you can use SQUASH to rebuild the diskpack with a valid label. Use the SQP command with the syntax *VALUE 0 E.*

Recovering to a Different Spindle

If a physical diskpack spindle is defective, and you have a recovery tape for the diskpack, you might be able to recover the diskpack to another spindle. This can be done by using the SQP command with the syntax *VALUE 0 E.* This causes SQUASH to ignore the diskpack serial number when it rebuilds the diskpack.

Caution

Use only a spare (scratch, unneeded) diskpack to perform a recovery to a different spindle, because the data on the diskpack will be overwritten and the spindle will inherit the data and attributes of the spindle that was used to create the recovery tape. The spare pack must have a capacity at least as large as that of the spindle used to make the recovery tape.

Interrupting SQUASH

Caution

Never use the DS command to stop the SQUASH utility.

You can safely interrupt SQUASH during its operation. In this way you can do a partial squash, completing it as time permits.

To safely interrupt SQUASH during operation, enter the following command:

```
<mix number of SQUASH program> QT
```

The command is safe to use at any time because it forces SQUASH to terminate the current phase at its next logical break point and then go to end-of-job. If the command is entered during the squash phase, SQUASH interrupts the squash phase at a logical break point and then proceeds through the complement phase before terminating.

Entering the QT command during the loading of a recovery tape leaves the diskpack in a "tape recovery" state because the data has been only partially restored to the diskpack; in this state the diskpack cannot be made ready. In this case use the SQP command, which causes the diskpack to be automatically reloaded from the recovery tape so that squashing and recovery can complete.

Monitoring the Squash Process

SQUASH provides information on its process. It always reports on errors it encounters. By setting programmatic switches 1 and 4, you can get additional details. Switch 1 produces an estimate of the percentage of squashing that has been completed. Switch 4 initiates reporting on the stage of squash processing.

Monitoring the Stage of Processing

The SQUASH utility goes through several phases to accomplish the squashing of the data on the diskpack. You can monitor the progress of the utility as it moves from phase to phase. There are two ways to accomplish this:

- Set programmatic switch 4, at program address 4, to a value of 1 when initiating SQUASH. This can be accomplished by using the following syntax:

  ```
  SQP <cc/u>; IN 4 1 1
  ```

- Set programmatic switch 4 to a value of 1 while the SQUASH program is executing. This is accomplished by using the following syntax:

  ```
  <mix number of SQUASH program> SW4 = 1
  ```

Either of these methods of setting switch 4 results in messages being displayed on the OCS showing the progress of the SQUASH utility.

SQUASH moves through the following phases:

1. Analyzing the contents of the diskpack
2. Creating or loading the recovery tape
3. Comparing the recovery tape to the data on the diskpack
4. Squashing the data on the diskpack
5. Complementing the diskpack and building a "working available table" on the diskpack

Messages for Stage of Processing

The following messages are produced when programmatic switch 4 is set. These messages are for informative purposes only, to let you know when SQUASH moves from one phase to the next.

During initial diskpack open:

THIS IS A <pack type> PACK WITH BINARY ADDRESSING.
THIS IS A <pack type> PACK WITH DECIMAL ADDRESSING.
THIS IS A <pack type> BASE PACK WITH BINARY ADDRESSING.
THIS IS A <pack type> BASE PACK WITH DECIMAL ADDRESSING.

While the diskpack contents are being analyzed:

ANALYSIS FILE TARGET ADDRESSES ASSIGNED.
ANALYSIS DIRECTORY/HEADER RECORDS UPDATED.
ANALYSIS FILE RECORDS UPDATED.
ANALYSIS COMPLETE.

While a recovery tape is being created:

START OF TAPE DUMP.
END OF TAPE DUMP.

While the recovery tape's contents are compared to the diskpack:

START OF PACK DATA COMPARE.
END OF PACK DATA COMPARE.

While the actual squash is being done:

START OF SQUASH PHASE.
END OF SQUASH PHASE.
BASE PACK READS: <number of reads performed from base pack>
BASE PACK WRITES: <number of writes performed to base pack>
PACK READS: <number of reads performed>
PACK WRITES: <number of writes performed>
DIRECTORY READS: <number of reads performed from pack directory>

While the disk available table is being built:

START OF COMPLEMENT PHASE.
END OF COMPLEMENT ANALYSIS.

When a bad I/O must be retried and eventually completes successfully:

SUCCESSFUL RECOVERY.

When a recovery tape is being loaded to the diskpack:

START OF PACK LOAD.
END OF TAPE LOAD.

Monitoring the Percentage of Processing

When SQUASH is in the squash phase of processing, which is the most time-consuming phase, you can request an estimate of the percent of squash processing that is completed. Use the following syntax:

```
<mix number of SQUASH program> SW 1 = 1
```

This is a rough estimate because it is dependent on many factors including the amount of checkerboarding on the diskpack. The estimate becomes more accurate as the percentage completed increases.

The estimate messages have the following form:

```
SQUASH IS APPROXIMATELY nnn% COMPLETE
ESTIMATED TIME TO COMPLETION IS hh:mm:ss (EOJ AT hh:mm)
```

Errors during SQUASH Operation

This section discusses errors that can occur while SQUASH is executing.

Errors on a Diskpack of Version 2 or Greater

If an error occurs on a base diskpack of version 2 or greater, the base diskpack must be recovered before squashing any continuation diskpacks in its family. The reason is that SQUASH uses the base diskpack when it squashes a continuation diskpack.

If an error occurs on a continuation diskpack of version 2 or greater, the continuation diskpack must be recovered before squashing any other diskpacks in its family. The reason is that SQUASH updates structures on the base diskpack during recovery of the continuation diskpack. If another continuation diskpack were squashed, the structures on the base diskpack would change, which would endanger the information that SQUASH needs to recover the failed continuation pack.

Error Display

If the SQUASH utility encounters an error during operation, it displays the following information:

- The structure on which the error occurred (directory, available table, file header, file area, etc.)
- The address and, if applicable, the file id on which the error occurred

In most cases, if the squash cannot finish successfully the recovery flag in the diskpack label is cleared so that the diskpack can be made ready and recovery measures can be taken.

Errors Messages and Error Recovery

This section is organized according to the type of error. Within an error type, the specific error messages, their meaning, and the recommended recovery actions are given. In general, the recommendations for recovery are listed in order from least severe to most severe. The recovery technique listed first should be tried first, if applicable. If that technique fails, try the technique listed second, and so on.

Tape Recovery Errors

```
RECOVERY TAPE IN INVALID FORMAT
RECOVERY TAPE IS INVALID VERSION
```

A recovery tape is in an invalid format or contains invalid data so it cannot be used to reload the diskpack.

- Use another tape drive.
- Use another recovery tape to reload the diskpack.

```
RECOVERY TAPE IS NOT FOR THIS PACK
RECOVERY TAPE IS NOT FOR THIS TYPE
```

A recovery tape's header record does not match the label information and/or the physical diskpack type for the diskpack to which it is being reloaded, so it cannot be used to recover the diskpack.

- Use another tape drive.
- Use the same diskpack spindle that created the recovery tape.
- Use a different recovery tape because the wrong recovery tape is being used to reload the diskpack.
- If the correct recovery tape is being used for a diskpack, use the VALUE 0 = E option when executing the SQUASH utility to bypass the diskpack label validation when reloading the diskpack.

```
COMPARISON ERROR ON RECOVERY TAPE
```

The data contained on a recovery tape does not compare equally with the data on a diskpack. This error will only occur if a compare is done (a compare is done by default) when a recovery tape is created or loaded.

- If a recovery tape was in the process of being created it will be purged. No data is lost on the diskpack and the diskpack can be made ready. If there are multiple recovery tape reels then the entire set is invalid and should be purged manually. Another attempt can be made to create a recovery tape.

- If the diskpack was in the process of being reloaded, it is in a "tape recovery" state and cannot be made ready. The recovery options at this point are:
  - Execute the SQUASH utility which will reload the diskpack from a recovery tape automatically. If this fails, try a different tape drive.
  - If a tape compared successfully when it was created, then the data on it should be valid and the problem is probably caused by the diskpack. As a last resort, if the data on a tape must be accessed and there is an available diskpack spindle with equal or greater capacity than the diskpack spindle used to create the recovery tape, the data on the recovery tape can be loaded to it by executing the SQUASH utility with a VALUE 0 = E statement. Keep in mind that the diskpack will inherit the data and attributes of the structures on the recovery tape. So, for example, if the diskpack that the data is being loaded to has a larger capacity than that of the diskpack on the recovery tape, the available table that is loaded will not reflect the true capacity of the larger diskpack. If loading a recovery tape to a different diskpack spindle is successful, the problem is likely caused by the diskpack and diagnostics should be run against it to determine the nature of the problem.
  - For a version 1 diskpack, rebuild the diskpack using the DISPKV utility. Then recovery measures, such as using SYSTEM/COPY, must be taken to reload the data to the diskpack or family.
  - For a diskpack of version 2 or greater, rebuild the diskpack or family with either the ALTER NEW PACK, ALTER FAMILY PURGE, or ALTER FAMILY REBUILD commands. Recovery measures, such as using SYSTEM/COPY, must then be taken to reload the data to the diskpack or family.

SEQUENCE ERROR ON RECOVERY TAPE

In an attempt to reload a diskpack from a recovery tape, the data on the recovery tape was found to be in an improper sequence.

- Use another tape drive.
- If a diskpack can be made ready after this error occurs then another attempt can be made to reload it from a recovery tape or other recovery measures, such as using SYSTEM/COPY, can be taken to reload the data to the diskpack.
- If the diskpack was in the process of being reloaded, it is in a "tape recovery" state and cannot be made ready. The recovery options at this point are:
  - Execute the SQUASH utility which will reload the diskpack from a recovery tape automatically. If this fails, try a different tape drive.
  - For a version 1 diskpack, rebuild the diskpack using the DISPKV utility. Recovery measures, such as using SYSTEM/COPY, must then be taken to reload the data to the diskpack or family.
  - For a diskpack of version 2 or greater, rebuild the diskpack or family with either the ALTER NEW PACK, ALTER FAMILY PURGE, or ALTER FAMILY REBUILD commands. Recovery measures, such as using

SYSTEM/COPY, must then be taken to reload the data to the diskpack or family.

Pack Errors when Reloading from a Tape

```
BAD LABEL WRITE - SECTOR = <sector address>
BAD MASTER AVAILABLE TABLE WRITE - SECTOR = <sector address>
BAD AVAILABLE TABLE WRITE - SECTOR = <sector address>
BAD DIRECTORY WRITE - SECTOR = <sector address>
BAD HEADER WRITE - SECTOR = <sector address>
BAD FILE AREA WRITE - SECTOR = <sector address>
BAD WRITE - SECTOR = <sector address>
```

During a diskpack reload from a recovery tape, an error occurred on a diskpack structure. The diskpack will be in a "tape recovery" state and cannot be made ready.

- A tape recovery can be forced by executing the SQUASH utility with a *VALUE 0 = D* statement.
- If a tape compared successfully when it was created then the data on it should be valid and the problem is probably caused by the diskpack. As a last resort, if the data on a tape must be accessed and there is an available diskpack spindle with equal or greater capacity than the diskpack spindle used to create the recovery tape, the data on the recovery tape can be loaded to it by executing the SQUASH utility with a *VALUE 0 = E* statement. Keep in mind that the diskpack will inherit the data and attributes of the structures on the recovery tape. So, for example, if the diskpack that the data is being loaded to has a larger capacity than that of the diskpack on the recovery tape, the available table that is loaded will not reflect the true capacity of the larger diskpack. If loading a recovery tape to a different diskpack spindle is successful, the problem is likely caused by the diskpack and diagnostics should be run against it to determine the nature of the problem.
- For a version 1 diskpack, rebuild the diskpack using the DISPKV utility. Recovery measures, such as using SYSTEM/COPY, must then be taken to reload the data to the diskpack or family.
- For a diskpack of version 2 or greater, rebuild the diskpack using the ALTER NEW PACK, ALTER FAMILY PURGE, or ALTER FAMILY REBUILD commands. Recovery measures, such as using SYSTEM/COPY, must then be taken to reload the data to the diskpack or family.

Label Errors

```
BAD LABEL READ
BAD LABEL WRITE
BAD OR INCOMPATIBLE BASE PACK LABEL (diskpacks of version 2 and greater only)
```

The diskpack label is invalid and the diskpack is in a "mid-squash" state so it cannot be made ready.

- Force a diskpack complement by executing the SQUASH utility with a VALUE 0 = C statement.
- If a recovery tape exists, force a tape recovery by executing the SQUASH utility with a VALUE 0 = E statement, which bypasses the validation of the diskpack label.
- For a version 1 diskpack, rebuild the diskpack using the DISPKV utility. Recovery measures, such as using SYSTEM/COPY, must then be taken to reload the data to the diskpack or family.
- For a diskpack of version 2 or greater, rebuild the diskpack or family with either the ALTER NEW PACK, ALTER FAMILY PURGE, or ALTER FAMILY REBUILD commands. Recovery measures, such as using SYSTEM/COPY, must then be taken to reload the data to the diskpack or family.

```
INVALID RECOVERY SPECIFICATION IN PACK LABEL
```

An unknown value is contained in the recovery flag field in the diskpack label. This flag is used by the SQUASH utility to determine the type of recovery that it needs to perform.

Clear the recovery flag in the diskpack label via one the following measures:

- Force a diskpack complement by executing the SQUASH utility with a VALUE 0 = C statement.
- If a recovery tape exists, reload the diskpack by executing the SQUASH utility with a VALUE 0 = E statement, which bypasses the validation of the diskpack label.
- For a version 1 diskpack, rebuild the diskpack using the DISPKV utility. Recovery measures, such as using SYSTEM/COPY, must then be taken to reload the data to the diskpack or family.
- For a diskpack of version 2 or greater, rebuild the diskpack or family using the ALTER NEW PACK, ALTER FAMILY PURGE, or ALTER FAMILY REBUILD commands. Recovery measures, such as using SYSTEM/COPY, must then be taken to reload the data to the diskpack or family.

Master Available Table, Available Table, and Directory Errors

```
BAD TABLE READ
BAD TABLE MARKER
BAD DIRECTORY LINK
BAD DIRECTORY READ - SECTOR = <directory address>
BAD DIRECTORY WRITE - SECTOR = <directory address>
```

A critical structure such as a directory, a master available table, or an available table may be corrupted and the diskpack is unusable. The diskpack is in a "mid-squash" state and cannot be made ready.

- Force a diskpack complement by executing the SQUASH utility with a VALUE 0 = C statement.
- If a recovery tape exists, a tape recovery can be forced by executing the SQUASH utility with a VALUE 0 = D statement.
- For a version 1 diskpack, rebuild the diskpack using the DISPKV utility. Recovery measures, such as using SYSTEM/COPY, must then be taken to reload the data to the diskpack or family.
- For a diskpack of version 2 or greater, rebuild the diskpack or family with either the ALTER NEW PACK, ALTER FAMILY PURGE, or ALTER FAMILY REBUILD commands. Recovery measures, such as using SYSTEM/COPY, must then be taken to reload the data to the diskpack or family.

```
INSUFFICIENT SPACE FOR AVAILABLE TABLE
AREAS LOST
```

The SQUASH utility cannot find 10 sectors available to extend the available table during the complement phase. The available table will not reflect the entire available space on the diskpack, but no file data is lost and the diskpack can be made ready.

- Files must be removed on the diskpack to free up some space for the extension of the available table. A diskpack complement can then be forced by executing the SQUASH utility with a VALUE 0 = F statement.

Error Messages When Squashing a File Area

```
BAD FILE AREA READ - SECTOR = <area address>
BAD FILE AREA WRITE - SECTOR = <area address>
THE AREA COULD NOT BE MOVED - FILE ID = <file id>
```

An unsuccessful read or write of a file area occurred when attempting to move a file area.

In addition, if the new area location overlaps the previous area location then the area is probably corrupted (depending on how much of the data was actually moved) and the following error message is displayed:

```
AN AREA MAY BE CORRUPTED STARTING AT SEC <area address> FOR A
    LENGTH OF <length of possible corruption in sectors>
```

```
BAD HEADER READ - SECTOR = <header address>
BAD HEADER WRITE - SECTOR = <header address>
```

```
AREA ADR IN HDR NOT UPDATED - AREA ADR = <new header address>,
    FILE-ID = <file id>
```

The file area was successfully moved but an error occurred when attempting to update a file header with the new area address.

In addition, if the new area location overlaps the previous area location then the area is corrupted and the following error message is displayed:

```
AN AREA IS CORRUPTED STARTING AT SECTOR <area address>
    FOR A LENGTH OF <length of corruption in sectors>
```

- The diskpack is in a "complement required" state so it cannot be made ready. The following message is displayed by the SQUASH utility, and when the SQUASH utility is re-executed a complement is performed automatically:

  ```
  A PACK COMPLEMENT IS REQUIRED - RE-EXECUTE SQUASH
  ```

- If a subsequent diskpack complement is successful, the diskpack is made ready. If there are no corrupted file areas then the diskpack does not have to be recovered and no data is lost.
- If a subsequent complement fails, the diskpack will not have any available space and the recovery flag in the diskpack label is cleared so that the diskpack can be readied. An attempt should be made to dump the data off of the diskpack and then it can be recovered by one of the following measures:
  - If a recovery tape exists, force a tape recovery by executing the SQUASH utility with a VALUE 0 = D statement.
  - For a version 1 diskpack, rebuild the diskpack using the DISPKV utility. Recovery measures, such as using SYSTEM/COPY, must then be taken to reload the data to the diskpack or family.
  - For a diskpack of version 2 or greater, rebuild the diskpack using either the ALTER NEW PACK, ALTER FAMILY PURGE, or ALTER FAMILY REBUILD commands. Recovery measures, such as using SYSTEM/COPY, must then be taken to reload the data to the diskpack or family.
- If a file has a corrupted area it can be restored using one of the following measures:
  - The file can be removed and restored from a backup copy.
  - The area can be examined to determine the nature of the corruption and measures can be taken to restore the affected data.

Error Messages When Squashing a File Header

```
BAD HEADER READ - SECTOR = <header sector address>
BAD HEADER WRITE - SECTOR = <header sector address>
THE HEADER COULD NOT BE MOVED - FILE ID = <file id>
```

An unsuccessful read or write of a file header occurred when attempting to move the header.

If the new header location overlaps the previous header location, an attempt to rewrite the header at its previous location is attempted since the header address in the file's directory entry still points to the previous location. If the write is successful the header information should be valid, otherwise the header is corrupted and the following error message is displayed:

```
THE HEADER IS CORRUPTED STARTING AT SEC <header address>
FOR A LENGTH OF <length of corruption in sectors>
```

```
BAD DIRECTORY READ - SECTOR = <directory address>
BAD DIRECTORY WRITE - SECTOR = <directory address>
HDR ADR IN DIR NOT UPDATED - ADDRESS = <new area address>,
    FILE-ID = <file id>
```

The header data was successfully moved but an error occurred when attempting to update a file's new header address in the file's directory entry. If the new header location overlaps the previous header location then the header is corrupted and the following error message is displayed:

```
THE HEADER IS CORRUPTED STARTING AT SEC <header address>
    FOR A LENGTH OF <length of corruption in sectors>
```

- The diskpack is in a "complement required" state so it cannot be made ready. The following message is displayed by the SQUASH utility, and when the SQUASH utility is re-executed a complement is performed automatically:

  ```
  A PACK COMPLEMENT IS REQUIRED - RE-EXECUTE SQUASH
  ```

- If a subsequent complement is successful, the diskpack is made ready. If there are no corrupted file headers then the diskpack does not have to be recovered and no data is lost.
- If a subsequent complement fails, the diskpack will not have any available space and the recovery flag in the diskpack label is cleared so that the diskpack can be readied. An attempt should be made to dump the data off of the diskpack and then recover it by one of the following measures:
  - If a recovery tape exists, force a tape recovery by executing the SQUASH utility with a VALUE 0 = D statement.
  - For a version 1 diskpack, rebuild the diskpack using the DISPKV utility. Recovery measures, such as using SYSTEM/COPY, must then be taken to reload the data to the diskpack or family.
  - For a diskpack of version 2 or greater, rebuild the diskpack using the ALTER NEW PACK, ALTER FAMILY PURGE, or ALTER FAMILY REBUILD commands. Recovery measures, such as using SYSTEM/COPY, must then be taken to reload the data to the diskpack or family.
- If a file header is corrupted then the associated file is lost. At this point the file cannot be removed explicitly but its name can be changed and a new copy of the file can be created or reloaded to the diskpack family. In order to remove the file, the diskpack must be rebuilt using the DISPKV utility for

version 1 diskpacks or the ALTER commands for diskpacks of version 2 and greater.

Miscellaneous Errors

```
PACK I/O ERR <result descriptor> ADR <sector address>
```

An I/O error occurred when reading or writing to a diskpack. This error is displayed in addition to a more specific error message describing on which structure the error occurred.

The recovery from this error will vary depending on the severity of the error. Use the appropriate recovery method as described under the specific error message.

```
UNKNOWN PACK UNIT TYPE, SQUASH ABORTED
UNKNOWN BASE PACK UNIT TYPE - SQUASH ABORTED (version 2 and greater packs only)
```

The diskpack type is not recognized by the SQUASH utility so a squash cannot be performed on the diskpack.

- None. The diskpack cannot be squashed using the SQUASH utility.

```
MISSING OR DUPLICATE AREAS ON PACK <cc/u>
DUPLICATE AREAS ON PACK <cc/u>
```

A diskpack has missing or duplicate areas so a squash cannot be performed on the diskpack.

- Execute the DSKOUT utility with the *PS <cc/u>* = syntax to find all of the missing and duplicate areas.
- If there are only missing areas, execute the SQUASH utility with the complement option (VALUE 0 = F) and then re-execute the SQUASH utility to squash the diskpack as originally attempted.
- If there are duplicate areas, the diskpack cannot be squashed until the situation is resolved.

Section 38
LOADER—Train Printer Buffer File Generator

The MCP intrinsic LOADER reads a card deck labeled TABLES and creates a single-record train printer buffer file on disk. This file is used by the MCP to determine the correct train ID and for loading the train printer translate buffer.

Train printer buffer files are identified by a train file ID. Included in the buffer file are the train ID, the train size, and the character set size. The train file ID can be up to six characters in length.

The train ID is used by the MCP to validate the print train currently on a particular train printer. The train ID consists of a 2-digit number, ranging from 00 to 7F; for 1100 LPM train printers only, 40 must be added to the ID (for example a train ID of 10 used on an 1100 LPM printer becomes 50). Having the 4-bit ON in the train ID indicates to the MCP that an 1100 LPM train printer is being used.

Train size is the physical number of print positions on the train printer train. Allowed sizes are 192 and 288. The character set size refers to the number of characters in one train set, where a set can be 16, 18, 48, 64, 72, or 96 characters. The program repeats this set to build a complete train table. If all of the train sets are not identical, the complete train table may be specified by making the character set size the same as the train size and listing all of the characters in the train.

Note: *Utilities such as the MCP DEBUG trace assume a standard train on the printer. A non-standard train with these programs will result in garbage output. It is recommended that a non-standard train be used only to circumvent temporary train limitations.*

LOADER Deck

The MCP utility LOADER reads in a card deck labeled TABLES, and from it creates a train printer buffer file. Table 38–1 shows the format for the LOADER input deck.

Table 38–1. LOADER Card Deck

Card Number	Columns	Function
1		
	1–6	File name of buffer file to be created
	7–8	Train ID. Range is 00–7F. Add 40 for 1100 LPM printer.
	9–11	Train size. Valid sizes are 192 and 288.
	12–14	Character set size. Valid sizes are 16, 18, 48, 64, 72, and 96. This should match the number of characters punched in the remaining cards.
	15	Character to be printed to represent an Invalid character. Normally a question mark (?).
	16	Character to be printed to represent a blank character. Normally EBCDIC 40 (blank character).
2 through end (200 to 750 LPM printers)		
	1–80	The EBCDIC characters corresponding to the characters in the train set. The character in column 1 corresponds to the first position in the train set, the character in column 2 corresponds to the second position in the train set, and so on. Use all 80 columns and then go on to the next card, using as many cards as necessary. The total number of characters should equal the character set size given in columns 12–14 of card number 1. The character to be printed for an invalid character must be specified as part of the train set.
2 through end (1100 and 1500 LPM printers)		
	1–80	This is the same as for 200 to 750 LPM printers, except that the character in column 1 corresponds to the **second** position in the train set, the character in column 2 corresponds to the **third** position in the train set, and so on. The first character in the train set is punched in the last column used. This is required for hardware timing considerations.

Notes: *The LOADER utility cannot generate a buffer file for 2000 LPM printers. For 2000 LPM printers, use the standard bound printer buffer file PRN256.*

Example

```
? EX LOADER
? DATA TABLES
EBC18 41288018?
235689$@4"71.*-,?0
? END
```

This example is for an 1100 LPM printer with a 288-character train and an 18-character train set. Invalid characters are represented by a question mark (?). The actual train ID is 01; 40 has been added to arrive at the 1100 LPM printer designation of 41.

Train Printer Buffer Files

You use the LOADER to create custom train printer buffer files. Supplied with the MCP release are six standard train printer buffer files. These files are bound to the MCP and are identical in format to the files created by execution of the LOADER program.

File Format

Train printer buffer files are single area, single record disk files with the format shown in Table 38–2.

Table 38–2. Train Printer Buffer File Format on Disk

Digit Position	Function
0–3	Length of train in use. Must be 192 or 288.
4–5	Train ID
6–7	The character to be printed to represent an invalid character. Can be EBCDIC character 40 (blank character). Must also be declared as part of the train set.
8–9	For 200–750 LPM printers, the first character on the train. For 1100 or 1500 LPM printers, the second character on the train. For 1100 or 1500 LPM printers, the first character on the train becomes character N in the file.
10–N	Remaining characters in the train set.
N+1 (Last character)	Character to be used to represent the blank character.

Buffer Files Supplied

The standard train buffer files bound to the MCP are listed in Table 38–3.

Table 38–3. Bound Train Buffer Files

File Name	Contents
FUL96	A full 96-character set for 1100 and 1500 LPM printers
FUL72	A full 72-character set for 1100 and 1500 LPM printers
T75048	A full 48-character set for 750 LPM printers
T75064	A full 64-character set for 750 LPM printers
T75096	A full 96-character set for 750 LPM printers
PRN256	A 256-character EBCDIC translate table for all buffered printers

Error Messages

MISSING CARDS TO DESCRIBE TRAIN <train file ID>

Card file EOF was encountered before all of the cards required to describe the character set were read.

INVALID TRAIN FILE ID: <train file ID>

The specified name was not alphanumeric or contained embedded spaces.

INVALID TRAIN LENGTH: <train ID>, MUST BE 192 OR 28899

The train length specified in columns 0–3 must be 192 or 28899.

INVALID TRAIN ID: <train ID>

The train ID must be a hexadecimal number in the appropriate range. One of the following two messages will also occur:

750 LPM TRAIN ID MUST BE 00 TO 3F

1100 LPM OR 1500 LPM TRAIN ID MUST BE 40 TO 7F

WARNING: INVALID CHARACTER SET LENGTH <length>

This message is produced if the character set length is not 16, 18, 48, 64, 72, 96, 192, or 288. Execution continues if no other errors are detected.

INVALID CHARACTER: <character> NOT PRESENT ON TRAIN

The character specified to be printed as the invalid character must be present in the train set.

TRAIN IMAGE FILE <train file ID> GENERATED

The train file was successfully generated.

TRAIN IMAGE FILE <train file ID> NOT GENERATED

One or more errors were found and displayed.

NO INPUT CARDS

Empty input file.

TRAIN LENGTH NOT MULTIPLE OF CHARACTER SET

The character set length specified does not fit the specified train length.

If any of these error messages occur other than INVALID CHARACTER SET LENGTH, TRAIN LENGTH NOT MULTIPLE OF CHARACTER SET, or FILE GENERATED, the file currently being generated will be purged and the program will continue with the next input card.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Cut along dotted line

Tape

Do Not Staple

Please Fold and Fasten

Tape

Fold here

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 817 DETROIT, MI 48232

POSTAGE WILL BE PAID BY ADDRESSEE

ENTERPRISE SERVER DIVISION
PRODUCT INFORMATION
UNISYS CORPORATION
25725 JERONIMO ROAD
MISSION VIEJO CA 92691-9826

UNISYS

Help Us To Help You

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

O Addition O Deletion O Revision O Error

Publication title

Document number Date

Comments:

Name Telephone number ( )

Title Company

Address

City State ZIP code

41270000-100