

**Burroughs** 

**Series B 90  
B1800/B1900  
Computer  
Management  
Systems  
(CMS)  
System Software**

**OPERATION GUIDE**

COPYRIGHT © 1980, BURROUGHS MACHINES LIMITED, Hounslow, England  
COPYRIGHT © 1980, BURROUGHS CORPORATION, Detroit, Michigan 48232

PRICED ITEM

Burroughs believes that the software described in this manual is accurate and reliable, and much care has been taken in its preparation. However, no responsibility, financial or otherwise, can be accepted for any consequences arising out of the use of this material, including loss of profit, indirect, special, or consequential damages. There are no warranties which extend beyond the program specification.

The Customer should exercise care to assure that use of the software will be in full compliance with laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change. Revisions may be issued from time to time to advise of changes and/or additions.

*Correspondence regarding this document should be addressed directly to:*

**The Manager, Systems Software Support,  
Technical Information Organization,  
Burroughs Machines Ltd.,  
Cumbernauld G68 0BN,  
Glasgow, Scotland.**

## LIST OF EFFECTIVE PAGES

Page	Issue
Title	Original
ii thru iii	Original
iv	Blank
v thru viii	Original
1-1 thru 1-4	Original
2-1 thru 2-12	Original
3-1 thru 3-26	Original
4-1 thru 4-118	Original
5-1 thru 5-18	Original
6-1 thru 6-18	Original
7-1 thru 7-25	Original
7-26	Blank
8-1 thru 8-68	Original
10-1	Original
10-2	Blank
A-1 thru A-31	Original
A-32	Blank
B-1 thru B-18	Original
C-1 thru C-5	Original
C-6	Blank
D-1	Original
D-2	Blank





# TABLE OF CONTENTS

Section	Page	Section	Page
1	INTRODUCTION		
	The CMS Concept		1-1
	Software Release Levels		1-3
	Software Patches		1-3
	Software Support		1-3
	To the Reader		1-3
2	BASIC CMS OPERATION		
	Introduction		2-1
	Peripherals		2-1
	System and User Disks		2-2
	Disk Format		2-2
	Disk Initialization		2-2
	Disk Files		2-3
	Disk File Names		2-3
	Disk File Group Names		2-4
	Disk Directory		2-4
	Indexed Files		2-5
	Dual Pack Files		2-5
	Magnetic Tape File Names		2-7
	Printer Files		2-7
	Other Peripherals		2-7
	Programs		2-7
	Executing Programs		2-7
	Intrinsics		2-8
	Mix Numbers		2-9
	Output Messages		2-9
	Format Diagrams		2-10
	Railroad Diagrams		2-11
3	CMS-COMMON INTRINSICS		
	Introduction		3-1
	AD (Assign Peripheral Device)		3-2
	AX (Accept a message for a program)		3-3
	CL (Clear Peripheral)		3-4
	DC (Data Communications operator input)		3-5
	DP (Discontinue and Dump)		3-6
	DS (Discontinue Program)		3-7
	DT (Systems Date and Time)		3-8
	FD (Form Define)		3-10
	GO (Restart a Stopped Program)		3-11
	MX (Display Current Mix)		3-12
	OL (Request for Status Information of Peripherals)		3-16
	PG (Purge Tape)		3-18
	PO (Power Off a disk)		3-19
	PR (Assign Program Priority)		3-21
	RY (Ready a Peripheral)		3-22
	SF (Substitute Disk File)		3-23
	ST (Temporarily Suspend a Running Program)		3-24
	SV (Save Peripheral)		3-25
	VL (Vertical Format on Printer)		3-26
4	CMS-COMMON UTILITIES		
	Introduction		4-1
	SYS-SUPERUTL		4-1
	Logging		4-3
	Common Utility Output Messages		4-3
	ADD (Add Files from Library Tape to Disk)		4-5
	AMEND(Disk File Amending)		4-7
	CH (Change File Name(s))		4-9
	CHECKADUMP (Compare Library Tape with Disk)		4-11
	CHECK.DISK (Check all Sectors of a Disk)		4-13
	COMPARE (Compare Files)		4-14
	Additional Capabilities		4-15
	COPY (File Copy)		4-19
	Copying Keyfiles		4-19
	Additional Capabilities		4-20
	CP (Compute)		4-29
	CREATE (Create Disk File)		4-31
	DA (Disk Analysis)		4-36
	Disk Mode		4-36
	File Mode		4-38
	General Notes		4-38
	Abbreviations		4-39
	DD (Disk Dump)		4-41
	Store Function :		4-41
	Restore Function		4-41
	DUMP(Dump Files to Library Tape from Disk)		4-43
	ECMA.LD (Load/Dump of ECMA Tape Files)		4-45
	Compact Initiation		4-46
	FL (Display File Attributes on Self-Scan)		4-49
	FS (File Squash)		4-51
	ICMD (Industry Compatible Mini Disk Access):		4-53
	IR (Initiate Log Recall)		4-56
	KA (Analyze Disk Space Assignment)		4-57
	KEY.CHECK		4-60

# TABLE OF CONTENTS (Continued)

Section	Page	Section	Page
DKX (Disk Allocation Information)	4-63	Output Messages	5-11
LB (Look Back in Log)	4-64		
LD (Tape Library Utility)	4-65	6 COMPILATION FACILITIES	
LF (Look Forward in Log)	4-66	Introduction	6-1
LIST (File List)	4-67	To Initiate a Single Compilation	6-3
Additional Capabilities	4-67	Use of Macro Calls	6-8
LOAD (Load Library Tape Files to Disk)	4-73	Compiler Dollar Options	6-9
LOAD.VFU (Load Vertical Format Unit)	4-75	To Interrogate the Status of Compilations	6-10
LR (List Directory)	4-78	To Restart an Aborted Compilation	6-11
MODIFY (Program Code File Modification)	4-81	To Clear an Aborted Compilation	6-12
Interactive Mode	4-82	Zip Failures	6-13
File Attributes	4-82	Reserved Words	6-14
PB (List Printer Backup Files) Format 2	4-87	Error Messages	6-15
PD (Print Disk Directory)	4-89	Restarting Executing CO Versions	6-18
PL (Print Log Files)	4-91	7 NUMBERED SYSTEM SOFTWARE	
RM (Remove Files from Disk)	4-95	OUTPUT MESSAGES	
SQ (Squash Disk)	4-97	Introduction	7-1
General Guidelines	4-102	Events # 1-9	7-2
SYCOPY (Copy Library Tapes)	4-103	Software Information	7-2
TAPELR (List Library Tape Directory)	4-105	Events # 10-19	7-4
TAPEPD (Print Name of a Library Tape)	4-107	Software Suspensions	7-4
TL (Transfer Log Files)	4-108	Events # 20-40	7-7
UNLOAD (Unload Files from Disk to Library Tape)	4-111	Invalid Request on Class A or B Communicate to MCP	7-7
UPDATE (Disk File Update)	4-113	Events # 41-49	7-11
WL (What Log File)	4-116	Fatal Device Errors	7-11
XD (Delete Bad Disk Sectors)	4-117	Events # 50-69	7-13
5 THE SORT/MERGE		Load Failures	7-13
Introduction	5-1	Events # 70-99	7-16
General Features	5-1	System Errors	7-16
Invoking the SORT	5-2	Events # 100-169	7-17
The SORT Language	5-3	Program Errors	7-17
The File Statement	5-3	Events # 170-199	7-23
The Key Statement	5-4	Sort Exception Events	7-23
The User-Option Statement	5-5	8 B 90 DEPENDENT SYSTEM SOFTWARE	
Functional Description	5-8	Introduction	8-1
Regular Record Sort	5-8	Power On	8-1
Inplace Record Sort	5-8	CMS Bootstrap Mode	8-3
Keyfile Creation	5-9	A Note on Forcing System Initialization	8-3
Tagfile Creation	5-9	Stand Alone Utilities	8-4
Merge	5-9	Loading Stand-Alone Utilities	8-4
Details of Sort Keys	5-10	Functions Available	8-4
Deleted Records	5-10	Common SAU Output Messages	8-5
		A Note on Dual Pack Files	8-6
		CH (Change disk file name)	8-7

## TABLE OF CONTENTS (Continued)

Section	Page	Section	Page
CLEAN (Clean BSM Drive Read/Write Heads)	8-9	DUMPANALYSE (Analyze B 90 Program Dump Files)	8-52
COMPARE (Compare two Disk Files)	8-10	GEN.DUMPFL (Create Empty B 90 Memory Dump File)	8-54
COPY (Copy files disk to disk)	8-12	GT (General Trace)	8-55
Dual Pack Files	8-14	ND (New Density)	8-58
DISCOPY (Duplicate a BSMII Disk)	8-16	PATCHMAKER (Patch B 90 Machine-Code Object Program Files)	8-60
FE (Initialize MTR Disk)	8-17	PMB90 (Analyze B 90 Memory Dumps)	8-62
IN (Initialize a Disk)	8-19	Starting the Utility	8-62
LD (Load Disk)	8-21	Using the Utility	8-63
LS (List File Sizes)	8-23	Power Off	8-68
OL (Print Status of Drives)	8-24		
PDX (Print Disk Directories)	8-25	10 B 1800/B 1900 DEPENDENT SYSTEM SOFTWARE	10-1
PO (Power Off)	8-26	TO BE PROVIDED	
RF (Reformat Disk)	8-27		
RL (Relabel a Disk)	8-29		
RM (Remove Disk Files)	8-30		
WS (Warm Start)	8-32		
Loading the MCP	8-33		
Basic Operation under MCP Control	8-35		
D-Lights	8-35	APPENDIX	
MCP States	8-35	A COMPLETE RAILROAD DIAGRAMS	A-1
Mix Numbers	8-35	B EXAMPLES OF PRINTED UTILITY OUTPUT	B-1
Automatic Volume Recognition (AVR)	8-36	C GLOSSARY OF TECHNICAL TERMS	C-1
Console Keyboard Under MCP Control	8-36	D RELATED DOCUMENTATION	D-1
Interrupting the MCP	8-37		
Memory Dump to Cassette	8-38		
Memory Dump to Disk	8-39		
System Load Errors	8-40		
Errors under MCP Control	8-44		
B 90 Dependent Utilities	8-49		
CONFIGURER (Configure B 90 Software System)	8-50		

## LIST OF ILLUSTRATIONS

Figure	Page	Figure	Page
1-1	CMS Portability	1-1	
2-1	Physical Disk Structure	2-3	
2-2	Disk Directory Structure	2-5	
2-3	Indexed Files	2-6	
2-4	Dual-Pack Files	2-6	
2-5	Sample SPO List	2-10	
2-6	Railroad Diagram Sample 1	2-11	
2-7	Railroad Diagram Sample 2	2-11	
2-8	Railroad Diagram Sample 3	2-12	
4-1	Railroad Chart for Compare Utility	4-16	
4-2	Railroad Chart for Copy Utility (Sheet 1 of 2)	4-20	
4-2		Railroad Chart for Copy Utility (Sheet 2 of 2)	4-21
4-3		Railroad Chart for List Utility	4-68
4-4		Paper Vertical Format Tape	4-75
5-1		Regular Record Sort	5-14
5-2		Keyfile Creation	5-14
5-3		Tagfile Creation	5-15
5-4		File Merge	5-16
5-5		Multiple Key Sort	5-17
6-1		Operation of CO Utility	6-2
8-1		B 80 Coldstart and Warmstart	8-2

## LIST OF TABLES

Table	Page	Table	Page
4-1	Peripherals Required by CMS-Common Utilities	4-2	
4-2	File Attributes Accessible by Modify	4-85	
4-3	PPB Attributes Accessible by Modify	4-86	
4-4	Mnemonics for Device Attributes for Modify	4-86	
5-1	Sign Convention for Signed 8-Bit Alphanumeric Fields	5-17	
5-2		Sign Convention for Signed 4-Bit Numeric Fields	5-18
5-3		Sign Convention for Separate Sign Character with 8-Bit Alphanumeric Fields	5-18
6-1		Zip Failure Messages	6-13
6-2		CO Reserved Words	6-14
6-3		Error Messages from CO	6-15

# SECTION 1

## INTRODUCTION

### THE CMS CONCEPT

CMS (Computer Management System) software is a powerful set of software items designed to operate on a number of different hardware products.

To the user of an individual hardware product running CMS software, there is a well-defined operator interface and set of programming languages. The importance of CMS is that the same user may use a different hardware product running CMS software, and with the same languages. This portability eliminates major operator retraining between different CMS products. It also allows freedom of interchange of programs between hardware products, limited only by availability of hardware features. For example, a program may be developed and compiled on one system, and run on another. Also, because the compilers are also programs, there is portability of compilers between hardware systems as well. Data files are similarly transferable from one system to another. This portability is achieved by building on the "soft machine" concept. Refer to figure 1-1.

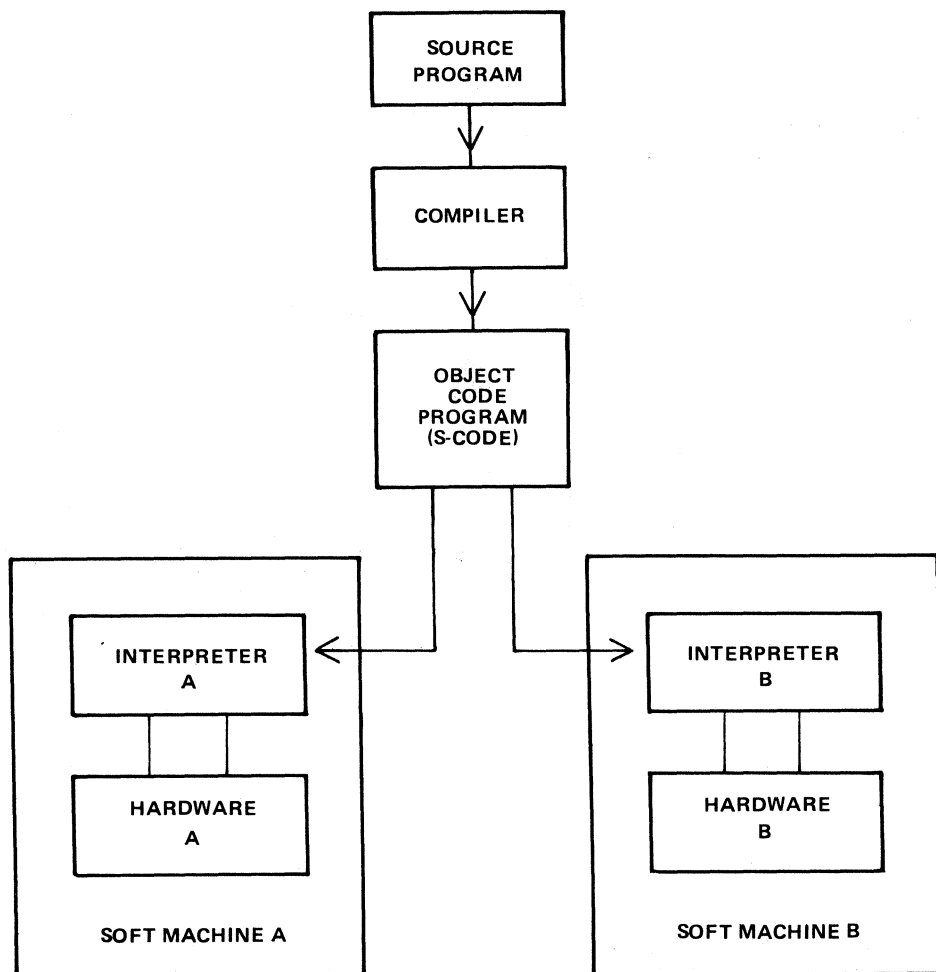


Figure 1-1. CMS Portability

The programmer writes a program in a high-level language. The CMS programming languages are:

COBOL

RPG (including RPGII)

MPL (CMS Message Processing Language)

NDL (Network Definition Language).

This program is written in 'source code'. This is then input to one of the CMS compilers which converts it to "object code" or "S-code". This is the executable program. The "S-code" is similar in design to the "machine code" of earlier generations of computer.

In earlier generations of computer this 'S-code' would be executed by hard-wired instructions. With the advent of fast micro-processor computers, however, it is possible to build a set of micro-instructions which interprets each "S-code" and executes it. The set of micro-instructions is therefore called an "interpreter". The combination of interpreter and micro-processor hardware is sometimes termed a "soft machine".

Now as the "S-code" is independent of any particular hardware, it is possible (and has been achieved in CMS) to build several soft machines which will execute a "object program" in a similar manner. Hence the CMS object programs are portable across the different CMS machines.

These machines include:

B 90

B 1800

B 1900

There are different CMS interpreters on each system. For example, on the B 90 the interpreters are:

BILINTERPX

COBOLINTX

NDL.INTERPX

BILINTERPX is used to execute programs written in MPL and in BIL (an implementation language used for compiler-writing which is so similar to MPL that they share the same S-code format). COBOLINTX is used to execute programs written in COBOL and RPG (these two languages share the same S-code format). NDL.INTERPX is used to interpret data communication controller programs written in NDL.

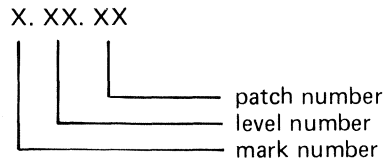
Certain common features needed in all programs (such as the handling of peripheral devices) have been collected together into a Master Control Program (MCP). The MCP is a micro-code program and is therefore specifically written for each hardware product. Thus there is a B 90 MCP, a B 1800 MCP and a B 1900 MCP. The MCP also controls the operator interface (which is standard across the CMS range) and maintains overall control of the system, providing complete resource management including multi-programming, I/O device handling and memory management.

CMS software also provides a number of utility programs. As these are written in MPL, they also are portable across the CMS range, limited only by hardware feature availability.

To cover the complete features of each CMS product line, certain aspects of the software are written for a specific product. These additional features include important operational characteristics, and are described in sections 8 through 10. Sections 2 through 7 of this manual cover items which are applicable to any CMS product.

## SOFTWARE RELEASE LEVELS

Each item on a CMS software release is identified by a three-part number, as follows:



The mark and level numbers constitute the release number. For example, the COBOL compiler 3.01.08 is the COBOL compiler included in the 3.01 release of system software, with patch number 08.

Software items from different releases should not be used together. For example, an interpreter from release 3.01 should not be used with an MCP from release 3.00.

This book describes system software relative to the 3.02 release.

### Software Patches

Within a particular release, patches to individual items may be issued. For example, an MCP identified by 3.01.12 contains certain improvements over an MCP identified by 3.01.11. A patch always increases the patch number. It is always advisable to use the highest patch versions within any one release. All system software items within a given release (mark and level numbers) may be used together, regardless of the patch number, unless explicitly stated otherwise at the time of release of the item.

Certain items may be patched by the user. The details are machine-dependent and are described in the relevant section (8 through 10).

## SOFTWARE SUPPORT

Throughout this book, suggestions are made for corrective action where possible, following a particular output message or symptom of failure. Sometimes the phrase “request technical assistance” has been used. This should be interpreted as a recommendation to contact your immediately higher support level if you are not sure of what to do or do not feel justified in attempting further action without competent advice.

All problems with the system should be recorded. This is for two purposes: to report the problem; and to avoid similar problems in the future. The report should contain the date and time and list the systems. As a minimum it is recommended that the SPO hard-copy printout or SPO log is kept for future reference.

## TO THE READER

This book is written as reference material. It is a guide to be consulted during operation of any CMS machine.

This book explains how to start and to stop the system software. As this is normally hardware-dependent, the relevant section (8 through 10) should be consulted.

Once the system software has started (that is, the system is under MCP control), the operator may interface with the MCP via the SPO (Supervisory Printout) device in order to execute programs. The type of device may vary with the hardware product, but input and output messages are standardized.

Section 2 of this book explains some general terms which should be understood in order to make full use of the CMS features. It explains how to cause programs to be executed. This section also explains how to read the diagrams used throughout the book to describe the format of input messages and other details.

Details of input messages are given, in alphabetical order, in sections 3 and 4. The items in section 4 are utility programs which are executed in the same manner as other programs. The items in section 3 are embedded features in the MCP. Refer to section 2 for a fuller explanation.

Sections 5 and 6 describe the sort/merge feature and the compilation feature respectively, and will be of special interest to programmers. Section 5 includes a functional description of the sort/merge feature.

Section 7 lists the messages which may be output to the SPO by the system software during execution of the system. As each message is identified on the SPO by a number, reference to this book can be made by this number.

For other items such as hardware and system software failures, refer to the particular hardware section (8 through 10) for details.



# SECTION 2

## BASIC CMS OPERATION

### INTRODUCTION

All CMS operation has two basic principles: it is disk-based; and operator communication is with the MCP by a SPO device. Other peripherals may be present, depending on the configuration. This section introduces some basic principles which should be understood by all CMS operators. The material in this section is common to all CMS products. Other details that are machine-dependent are given in the relevant section.

### PERIPHERALS

Each peripheral is referenced by a three-character abbreviation, where the first two characters give the type of peripheral and the third character refers to the particular peripheral by the letter A, B, and so on. For example, LP is the abbreviation for a line printer, so the first line printer is referred to as LPA, and the second is LPB.

The peripheral types are listed below:

- AC – console with any output device
- AM – any multi-function card unit
- AP – any (serial or line) printer
- AR – any card reader
- AT – any magnetic tape
- CP – any card punch
- CT – cassette tape
- DC – data communications controller
- DF – fixed disk
- DI – industry-compatible mini-disk (ICMD)
- DK – disk cartridge (any type of speed)
- DM – Burroughs super mini disk (BSMD and BSMDII)
- DP – disk pack
- KB – Keyboard
- LP – line printer
- MT – magnetic tape (reel)
- M8 – 80-column multi-function card unit
- M9 – 96-column multi-function card unit
- PC – console with serial printer
- P8 – 80-column card punch
- P9 – 96-column card punch
- R8 – 80-column card reader
- R9 – 96-column card reader
- RS – Reader Sorter
- RT – Real Time Clock

SC – console with SELF-SCAN<sup>®</sup> device  
SD – Screen Display  
SP – serial printer (on console)  
SS – SELF-SCAN<sup>®</sup> device

If the configuration contains more than one device of the same type, the designation (A, B, and so on) depends on the location of the peripheral controller in the hardware. If there is only one dual-drive cartridge controller, the upper drive is DKA and the lower drive is DKB. If there is only one dual-drive Burroughs super-mini-disk controller (for example, on a small B 80 with in-built mini disk), the upper drive is DMA and the lower drive is DMB.

The three-character references are used in all operator communication with the MCP (refer to section 3).

## SYSTEM AND USER DISKS

The MCP resides on a disk unit. At warmstart time (when the system is started up and the MCP begins to function) the MCP notes the disk containing the executing MCP code. This is called the “system disk”.

During operation there is only one system disk. Other disks may contain a copy of the MCP code, but only the disk from which the MCP is running is the system disk.

All other disks on the system during machine operation are called “user disks”.

There is one restriction on the portability of system disks between different CMS products. A system disk may not be taken to a different CMS product and used there as a system disk. It may, however, be used on the second system as a user disk. It may also be used on the first system as a user disk. User disks may always be interchanged between different systems.

## DISK FORMAT

A disk consists of one or more platters, one or both surfaces of which may be used to record data. The recording area of disks is divided into the following physical items:

### Track:

An area of one surface of a disk which is at the same distance from the center of the disk. The entire track can be accessed without moving the position of the read/write head.

### Sector:

The basic unit of disk address, size 180 bytes on all Burroughs disks, and 128 bytes on ICMD. A physical read or write uses a complete sector. There are several additional bytes in each sector, used only by the hardware and not accessible to user programs. The sector is also called a “segment”.

### Cylinder:

If there is more than one surface, each track at the same distance from the center makes a cylinder. The entire cylinder may be accessed without altering the position of the read/write heads.

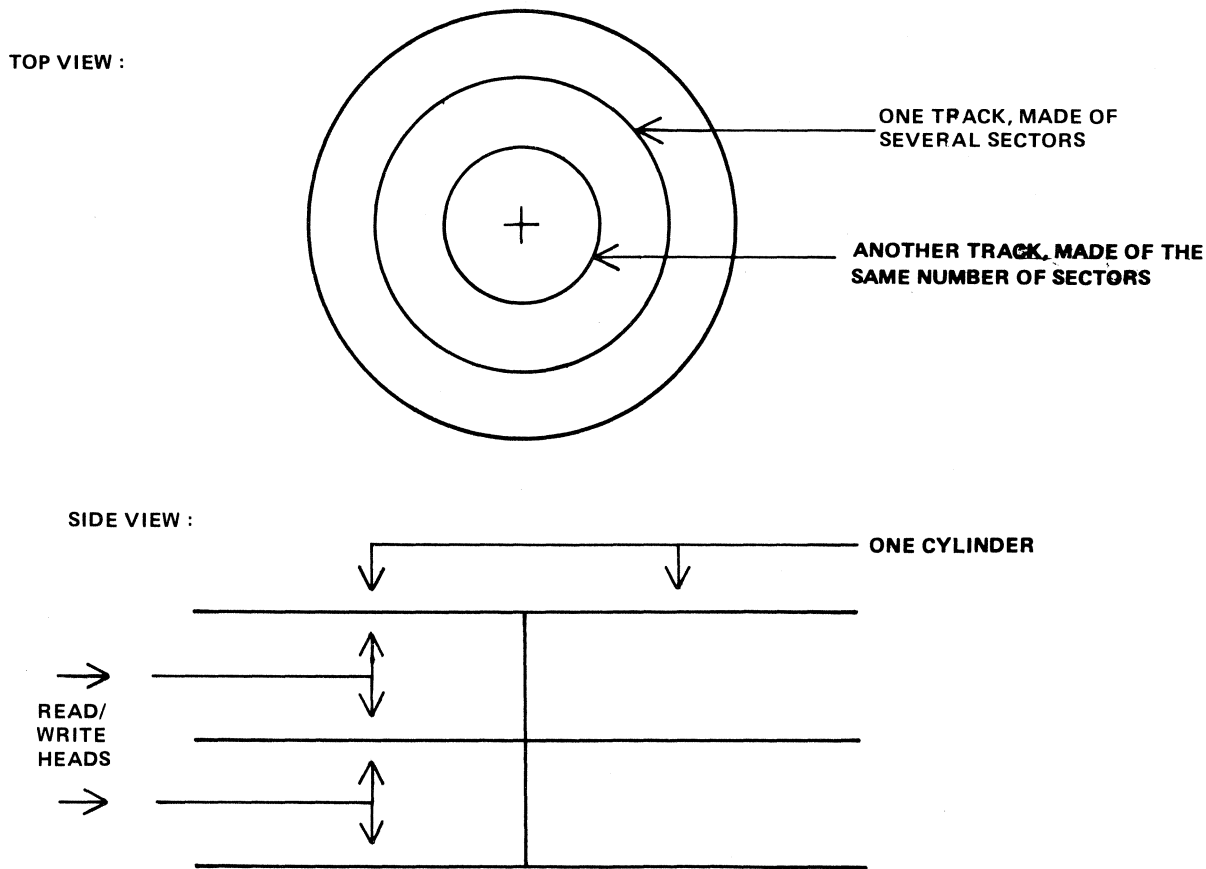
Figure 2-1 illustrates these terms.

## Disk Initialization

Each disk must be initialized before use on a CMS machine. Initialization creates correct sector addresses throughout the disk recording surface, then writes certain data in the low-address part of the disk. The first sector is numbered sector zero, and the first track is numbered track zero. A disk with a bad track cannot be initialized. The method of initializing the disk is machine-dependent refer to the appropriate section.

Sector zero contains the disk label. This includes the name of the disk, or “disk-id”. Every disk has a disk-name. This disk-name can be from one to seven characters, using the set A to Z, 0 to 9, and the dot (“.”) and hyphen (“-”).

SELF-SCAN<sup>®</sup> is a registered trademark of Burroughs Corporation.



**Figure 2-1. Physical Disk Structure**

## Disk Files

Information is stored on a disk in a “disk file”. There may be many files on one disk. Each file is referenced by a “file name”. A file name can be from one to twelve characters, using the set A to Z, 0 to 9, and the dot and hyphen. Each disk contains a directory of the files on that disk. This directory is accessed by utilities such as KA and PD (see section 4).

Information can be of different types: normal data, accessed by programs; special data, accessed by the MCP; and programs themselves. The MCP is itself a program, and so are other “system files” such as the interpreters. System files have special restrictions in that a control is placed on their removal (see RM section 4).

## Disk File Names

On any system, every disk file (whether data or a program) is accessed by a two part reference, as follows:  
 disk-name/file-name

For example, the disk file M101A/REP200 is a file with a file-name REP200 to be found on the disk with a disk-name M101A.

It is not necessary to give the name of the system disk when referring to files residing on the system disk. Alternatively, a disk-name of 0000000 by convention refers to the system disk. For example, the disk file REP200 or 0000000/REP200 is a file with a file-name REP200 to be found on the system disk.

It is not allowed to have two disks of the same disk-name in use at the same time. It is not allowed to have two files of the same file-name on the same disk. However, it is quite permissible for two different disks to contain a file with the same file-name. For example, the files M100A/REP200 and M101A/REP200 refer to two different disk files (although one may be a copy or update of the other).

## Disk File Group Names

In many utilities (see section 4) it is convenient to refer to groups of files, depending on common starting characters of their file-names.

All files on a disk may be referenced by the equals symbol (“=”). For example, the reference M101A/= refers to all files on the disk with disk-name M101A.

All files beginning with, say, the characters REP may be referenced by REP=. For example, the reference M101A/REP= refers to all files on disk M101A with file-names of REP200, REPA, REP678P, and so on.

In general, a group-name consists of an equals symbol (“=”) optionally preceded by up to ten symbols which are the first part of the file-names of each of the files in the group.

### *Example:*

Consider a disk M101A containing files with file-names:

PR200,REP100,REP200,REP250,RQ510,CRCOPY

Then the following group-names refer to the files indicated:

M101A/=

PR200, REP100, REP200, REP250, RQ510, CRCOPY

M101A/REP=

REP100, REP200, REP250

M101A/R=

REP100, REP200, REP250, RQ510

## Disk Directory

The disk directory is a table on every CMS-initialized disk which enables the MCP to locate any disk file by name. Full details of the directory layout are given in the CMS MCP manual.

The directory is a fixed size determined at disk initialization time, based on the maximum number of files to be placed on the disk. An attempt to create more files than there are entries in the directory will give an appropriate **MCP run-time error message**.

The directory consists of three parts:

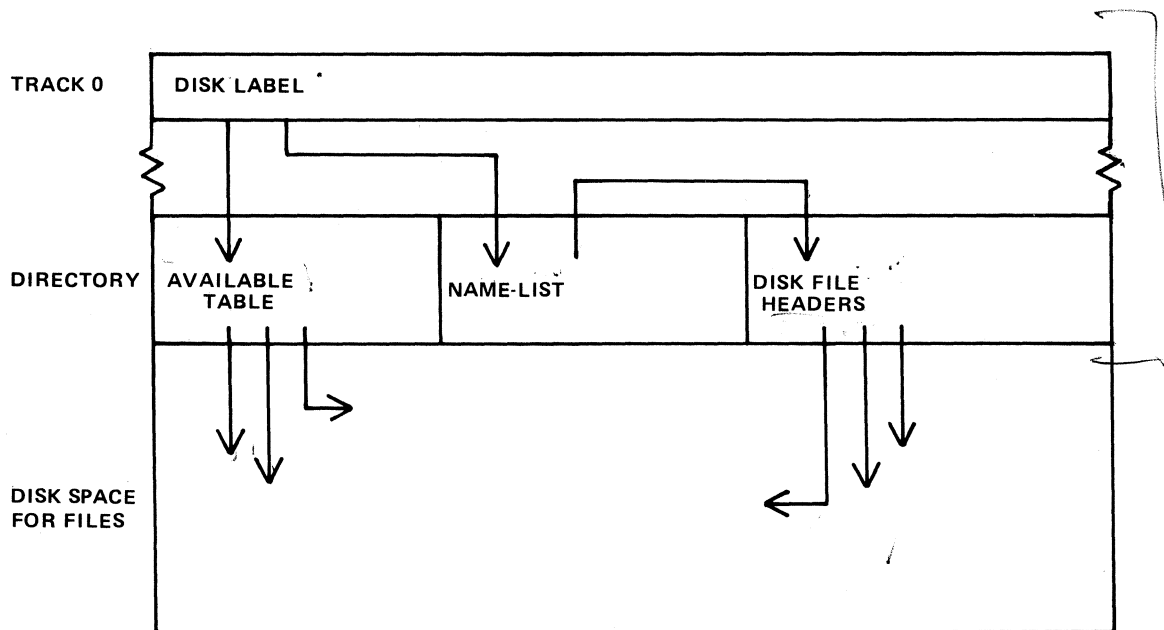
the name-list

the disk file headers for each file

the available table

The relationship between these parts are given in figure 2-2. The name-list is a list, by file-name, of each file existing on that disk. A search through this name-list will reveal if a file is present or not: if present, the name-list entry points to the disk file header for the file. This is a table giving the location of each part of actual data in the file (the file may be divided into up to sixteen separate physical areas on the disk). In the figure only one area is indicated. The available table is a list of the disk areas not in use by a file. When a new disk file is created, an available space is found from this table and an entry made in the name-list, then the space is used to write the file information. When a disk file is removed its entry is deleted from the name-list and the areas specified in the disk file header are entered in the available table.

If there is insufficient space on a disk to allocate new disk file areas, a “NO USER DISK” message is given by the MCP. The operator may remove a file (see RM) to make more space available. The KA utility (see section 4) and KX function provide information on the available space on a disk.



**Figure 2-2. Disk Directory Structure**

As a simplification, it may be stated that when a disk is initialized the directory is rebuilt with no entries, indicating that the entire disk space is available apart from the directory itself. In fact, any bad areas on the disk are marked in the directory so that they cannot be allocated to files (see also the XD utility); also, there is a special entry called "SYSTEMEM" which enables certain programs such as PD and RM (which access the directory) to operate successfully.

## Indexed Files

Indexed files are in fact a pair of files, the "key file" and the "data file". They may reside on the same or separate disks. Each file in the pair has a separate entry in the disk directory of the disk on which it resides. A special table at the beginning of the key file (the "key file parameter block") gives, among other information, the disk-name and file-name of the associated data file. See figure 2-3 for a diagram of the relationships between the two files.

The purpose of indexed files is to simplify access to data in the data file by using a set of keys (such as account number) in each record of the data file. These keys are placed in the key file. A key file may be created by the SORT utility and intrinsic (see section 5, where examples are given).

Special consideration must be given to copying indexed files, due to the link between the key file and data file. This is especially true when copying from one disk to another. Details are given in each relevant section (see COPY utility, section 4; also the machine-dependent copy facilities).

## Dual Pack Files

As mentioned before, a disk file may be divided into up to sixteen separate areas. If these areas are located on two separate disks the file is known as a "dual pack file". Such files may be created by the AD intrinsic in response to a "NO USER DISK" message (see section 3).

There is an entry in the directories of both disks for a dual pack file, together with the disk-name of the other disk. Each disk directory has a copy of the disk file header for this file, but the table of locations for each file area also indicates if the area is located on "this" disk or the "other" disk. This is shown diagrammatically for a file with four areas in figure 2-4. In most applications it is necessary for both disks of a dual-pack file to be on-line at the same time.

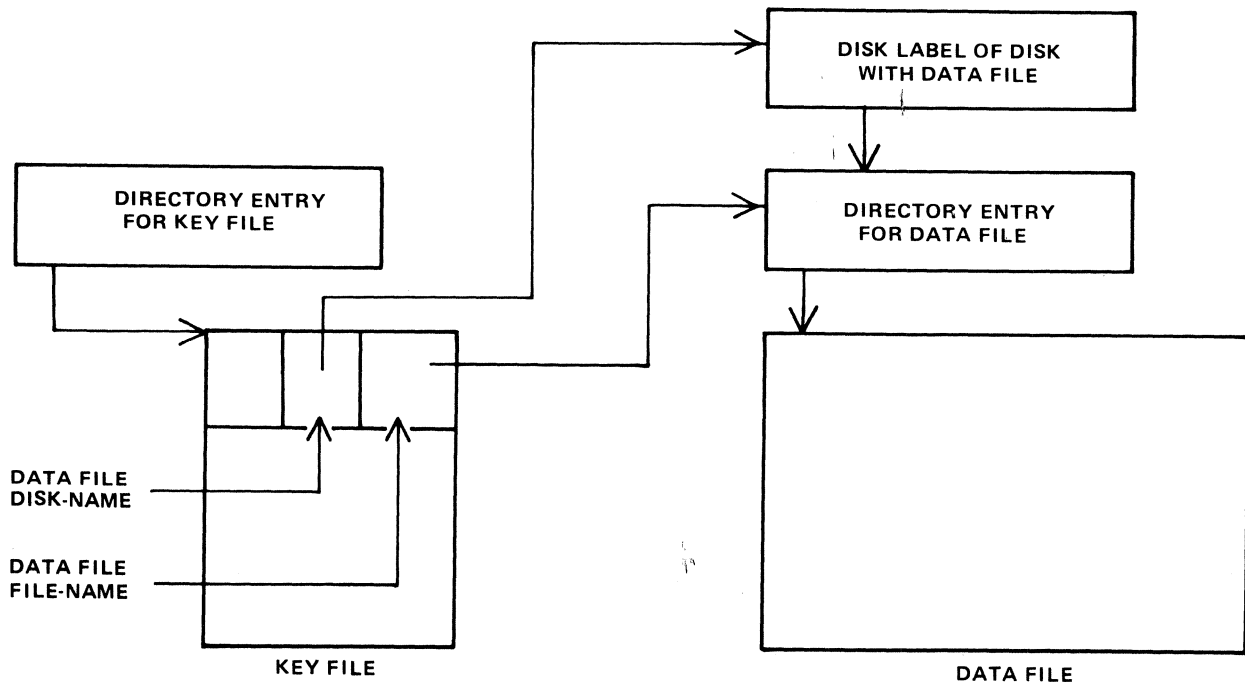


Figure 2-3. Indexed Files

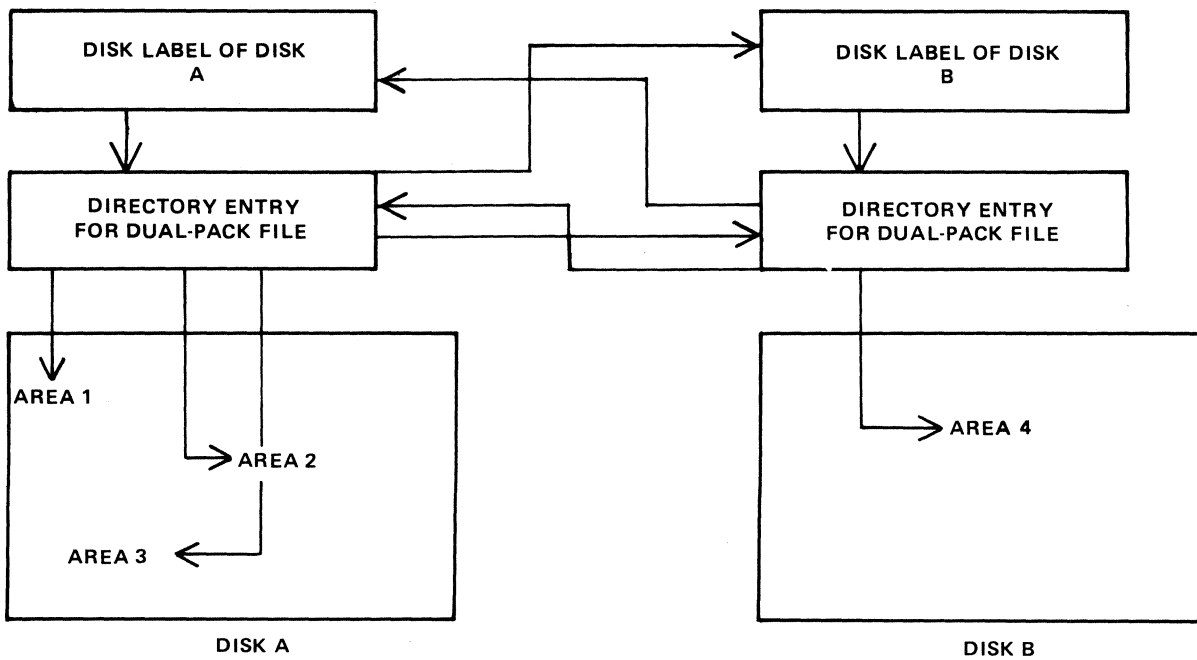


Figure 2-4. Dual-Pack Files

## MAGNETIC TAPE FILE NAMES

Note: this includes tape cassette.

A tape may be used to store data either on one file (a “single-file tape”) or as a “multifile tape”. Each file is separated by a tape mark. Additionally, each file normally has a beginning and an ending label. A multifile tape has also a special beginning (“volume”) label.

On loading a tape, the MCP reads the first label to determine the tape name. Tape file names are in two parts:

multifile-name/file-name

For a single-file tape, the multifile-name will be “0000000”. The format of the multifile-name is the same as for the disk-name of a disk file.

The COPY utility (section 4) produces a single-file tape when copying to tape. The LD utility (section 4) always produces multifile tapes called “library tapes”. Library tapes are referenced by the multifile-name: there is a standard convention for labelling all the files on a library tape. For full details of tape formats, refer to the CMS MCP manual.

Tapes (multi-file or single-file) may be unlabelled. Such tapes must always be accessed via the AD intrinsic (section 3) because there is no label that the MCP can recognize when the tape is loaded. Tapes containing labels that are non-standard are also treated as multifile unlabelled tapes.

## PRINTER FILES

There are two types of printer: a wide line printer and a console printer, depending on available hardware. The console printer is also known as a “serial printer”. These hardware devices are also referred to as “files” and are given file-names of up to seven characters. When the file is opened and closed, an identifying print line is given to indicate the name of the file. This file-name is also used in MCP messages. Refer to the CMS MCP manual for full details.

It is possible to designate a file type of “any printer”. Such a file will be written to a wide line printer if this peripheral is available. If not available, this file will be written to the console printer if available. If there is no console printer either, the MCP will display a “NO FILE” or “DEVICE REQUIRED” message.

## OTHER PERIPHERALS

All peripherals are treated as files for input, output or a combination of input/output, depending on the hardware type. The use of any peripheral device is governed by the file-name of up to seven characters, which will appear in any related MCP messages. Refer to the OL intrinsic (section 3) for other details.

## PROGRAMS

An executable program is information stored on disk as a disk file. It is referenced in the same way as any data file: that is, through the disk-name and file-name (or just the file-name if the program resides on the system disk). The rules for the program name are the same as for any disk file name.

A “utility” is a program provided for general use by all CMS operators, for house-keeping and other general purposes. For example, the LD utility enables operators to load and dump disk files from disk to magnetic tape for backup purposes.

### Executing Programs

In order to execute a program, part or all of the information in the disk file must be brought into memory and placed under control of the MCP. This is called “program load”, and takes a certain interval of time.

Programs may be loaded and executed by merely providing the name of program file to the MCP. If so desired, the keyword "EX" may be placed before the program name. For example, suppose one wishes to execute a program that resides on a disk PR200A in a file called DCS. Either the input

```
EX PR200A/DCS
```

or just

```
PR200A/DCS
```

will cause the program to be loaded and executed.

Depending on the system, a BOJ (beginning-of-job) message may be displayed by the MCP after the program has been loaded, and a EOJ (end-of-job) message may be displayed by the MCP at the end of the program. The display of these messages may be turned on or off for individual programs by the MODIFY utility (see section 4).

Failures may occur when attempting to load a program. For example, the requested program may not be on disk. A list of load failure messages is given in section 7.

Many programs enable the operator to enter further information after the program name. This is known as an "initiating message" and the contents are entirely dependent on the program. Nearly all the utilities in section 4 allow further information, the format of which is given in the description of each utility program. For example, the input

```
COPY REP202 TO RPTAPE
```

consists of the command to load and execute the program called "COPY" (found on the system disk in this example), followed by the information "REP202 TO RPTAPE" which is passed to the program. There are two types of error which can be made: either there is a load failure (because, for example, the COPY program is not on the system disk), when the MCP would issue an appropriate message; or the following information is an incorrect format for the program, when the program itself would issue a message. In the former case, the MCP message is described in section 7. In the latter case, the output message is described under each utility.

Note that if the utility resides on, say, the disk PR2, the input message would be

```
PR2/COPY REP202 TO RPTAPE
```

or

```
EX PR2/COPY REP202 TO RPTAPE
```

In section 4 this additional information is omitted in the interest of clarity. It is, however, common for utilities to reside on a disk other than the system disk, in which case the disk-name must be provided.

It is also possible for programs to be automatically executed by another program. In this case, the first program is said to "zip" the second program. No operator input is used in this case, but the BOJ message may be displayed for the zipped program.

## INTRINSICS

There is an important type of operator input that does not involve a command to execute programs or utilities. These messages are calls on "intrinsic" which are part of the MCP. Those intrinsic which are common to all CMS machines are described in section 3. Other intrinsic are given in the relevant machine-dependent section.

Because an intrinsic is part of the MCP, there is no separate program corresponding to the name of the intrinsic. Therefore the keyword "EX" is not allowed in a call on an intrinsic, neither can a user disk-name be specified. There is no program load time because the MCP is already executing. For example, the input

```
RY DMA
```

is a request to the MCP to ready (RY) the disk peripheral designated by DMA. This input message to the MCP must not be preceded by the keyword "EX".



## MIX NUMBERS

As a program is loaded, the MCP assigns it a number from its table of executing tasks. This is the “mix-number” and is used in any messages output by MCP relating to this task. The mix-number is also used in all messages input by the operator for this task. Some input messages also require the corresponding program name as well as the mix-number. The MX intrinsic (see section 3) may be used to determine the current mix of tasks.

The allocation of mix-numbers is dependent on the CMS product. Refer to the corresponding section for more details.

## OUTPUT MESSAGES

As mentioned earlier, messages may be output on the SPO either by the MCP and other system software or by the program. It is important to distinguish between the two types of output messages in order to look up the message in the appropriate place.

Messages output by the MCP are of two kinds: short responses to intrinsics, and longer descriptions of any event to be brought to the attention of the operator. The short descriptions are self-explanatory: for example, the input message

OL LPA

(an intrinsic to inquire of the status of line printer LPA) may result in the response

LPA READY

Similarly, the short message

LPA NOT READY

will be displayed if LPA is stopped by the operator or through any fault. The longer descriptions are always referenced by an “event number” enclosed in brackets. The format of these messages is given in section 7, and operators should be generally able to recognize that such a message has been output by the MCP.

For example, the message

10/LIST <17> WAITING UNLAB LISTPRT AP NO FILE

indicates an MCP message with event number 17, and reference should be made to section 7 for information on possible causes and suggested actions to take.

Messages with event numbers may also be output by other parts of the system software such as interpreters and the sort-intrinsic, although the overall format is similar. After recognizing the event number, reference should be made to section 7 (or section 5 for sort-related messages).

Messages output by all other programs are known as “displays” and may be preceded by the keyword “DISP”. Note, however, that utility programs may display messages without this preceding keyword.

All messages output by the utility programs described in this manual are listed under the respective utility. For example, messages displayed by COPY utility are listed under the COPY utility. Messages may additionally be displayed by the MCP for events related to the execution of the COPY utility (for example, if the COPY utility needs space on a particular disk, a “NO USER DISK” message will be output) but these MCP messages will always be distinguished by the event number.

Messages displayed by other programs are not discussed in this manual. Reference must be made to the appropriate manual or operating instructions for that program.

Figure 2-5 illustrates a sample SPO list giving a mixture of messages described in this section. Note in this example that the utility programs LIST and LR do not give rise to BOJ and EOJ or DISP messages. The user program PROGA shows all three messages. These messages may be turned on for utilities by using the MODIFY utility (section 4).

```

input command to run LIST --> LIST COLLETTE
MCP output message event 10-> 01/LIST <10> WAITING COLLETTE DK NO FILE
input command to run PROGA -> PROGA
MCP message for PROGA BOJ --> 02/PROGA BOJ PR IS A
input command to run LR ----> LR =
next line is PROGA display -> 02/PROGA DISP:
actual display information -> PROGRAM A VERSION 3.01.05
input request OL intrinsic -> OL LPA
MCP response to OL message -> LPA LRPRINT IN USE BY 03/LR
input request MX intrinsic -> MX
MCP response to MX message -> 01/LIST SUSPENDED WAITING ON NO FILE
-> ...CONDITION
-> 02/PROGA A EXECUTING
-> 03/LR B EXECUTING
MCP message for PROGA EOJ --> 02/PROGA ECJ
input request ST intrinsic -> ST 3
MCP response to ST message -> 03/LR STOPPED

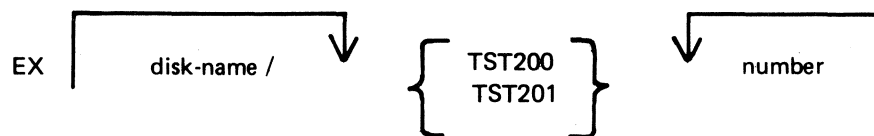
```

Figure 2-5. Sample SPO List

## FORMAT DIAGRAMS

Most of the descriptions of input messages in this book are given as simple format diagrams with corresponding descriptive text and examples. An example will illustrate how to read such format diagrams.

*Example:*



In this format, items in lower-case (“disk-name” and “number” in this example) are to be replaced by actual values (such as “PR2” and “27”). Other items are included in the input message as they are found. Spaces are required whenever necessary to avoid ambiguity. In the example, it is not strictly necessary to separate the disk-name and the slash (“/”) with a space because the slash cannot be part of the disk-name according to the rules for disk-names. Extra spaces may however, be added for legibility. If an arrow in the left-to-right direction is encountered, the items under the arrow may be omitted. Curly brackets are used to denote alternatives. The alternatives are placed in a list underneath each other. (Each alternative item may be more complex than the example quoted: it may contain optional parts and further alternatives). If an arrow in the right-to-left direction is encountered, one may return to the point underneath the arrow and continue building up a valid input message. In the example quoted, after adding a valid number (say “27”) one may return to add a second number (say “52”). In fact, the format diagram does not specify how many times one may continue to do this, but details are given in the text.

Here are several valid input messages which can be generated from the example. (Note that a disk-name can consist of up to seven characters, see earlier):

```

EX TST200 57
EX TST201 259
EX PR2/TST200 36
EX PR2/TST200 2 52 574 361
EX M101A/TST201 1 2

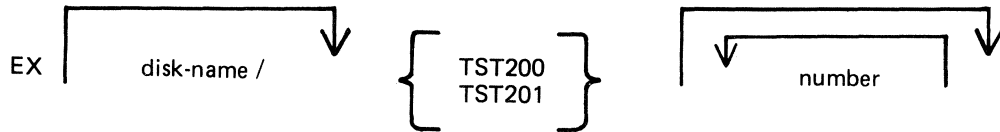
```

Here are several invalid input messages according to the example:

```
EX PR2/TST200
EX PR2 TST200 36
EX TST202 36
TST201 259
EX PR2/M101A/TST201 1 2
```

Here is a slightly more complicated example, which makes the number or list of numbers optional:

Example:



The input messages

```
EX PR2/TST200
EX PR2/TST200 56
EX PR2/TST200 27 56
```

are now all valid.

These simple format diagrams are easy to understand in conjunction with descriptive text and examples, but cannot be used if the format becomes too complex. In the latter case a rigorous notation known as “railroad diagrams” is employed (see below). In some case in the text of this book, the format has been deliberately simplified for the sake of clarity, with further details given in the text. More complex features have been described by railroad diagrams (see, for example, the COPY and LIST utilities in section 4). Appendix B gives complete railroad diagrams as a handy reference for those who need the exact definition of any input message.

## RAILROAD DIAGRAMS

The equivalent railroad diagram to the first format diagram is given in figure 2-6.

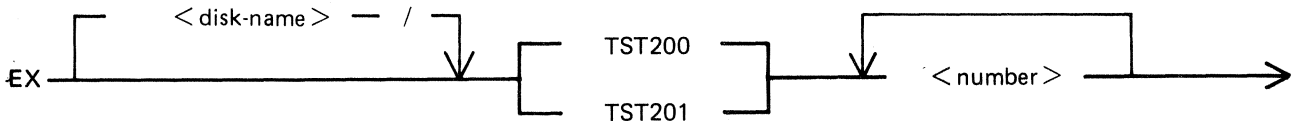


Figure 2-6. Railroad Diagram Sample 1

To form valid input, follow the railroad “track” from left to right or in the direction of the arrows. A junction in the track indicates that alternative paths may be followed. Items enclosed in angled brackets “<” and “>” must be replaced with actual values, as before. Each item not enclosed in angled brackets is included as it is found. Spaces are added where necessary, as in format diagrams.

The equivalent railroad diagram to the second format diagram is given in figure 2-7.

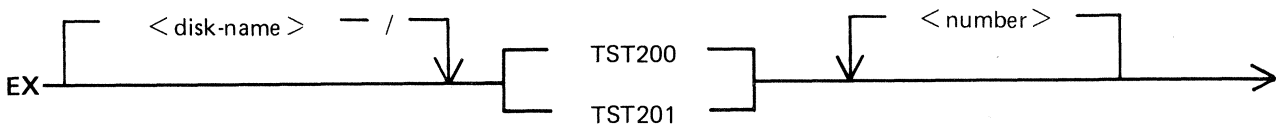
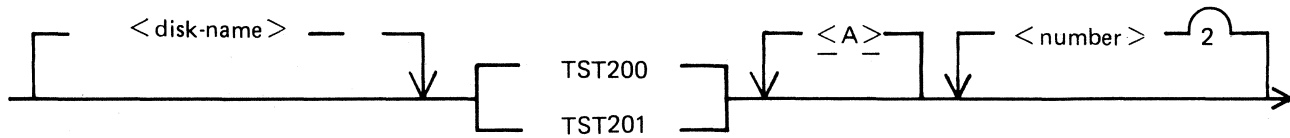


Figure 2-7. Railroad Diagram Sample 2

There are two other features available in railroad diagrams to make possible the exact specification of any input message. These are illustrated in figure 2-8. Firstly, the maximum number of times around a loop may be controlled by including the number



**Figure 2-8. Railroad Diagram Sample 3**

in the track of the loop. In the example, it is possible to omit the <number>, or to include either one or two values of <number>. Secondly, if angled brackets are to be included as part of the message, these must be underlined. In the example, there is an optional part of the message which consists of the three characters "<A>". The following messages would then be valid:

```
EX PR2/TST200
EX PR2/TST200 27
EX PR2/TST201 27 56
EX PR2/TST201 <A>
EX PR2/TST200 <A> 56
```

but the following would be invalid:

```
EX PR2/TST200 27 56 243
EX PR2/TST201 A
EX PR2/TST201 A 73
```

Note also that if a number under a loop is preceded by an asterisk ("\*"), then that loop must be included in the syntax at least the number of times specified. For example, if the loop included the characters "\*1", then the loop must be included at least once.

# SECTION 3

## CMS-COMMON INTRINSICS

### INTRODUCTION

This section describes, in alphabetical order, those input commands which are embedded in ("intrinsic to") the MCP, and which are common to all CMS products.

As discussed in section 2, it is not valid to precede these messages with "EX", because the intrinsics are not separate programs to be loaded and executed. The intrinsics cannot be executed from a user disk, because by nature they are part of the MCP which is on the system disk.

The response to these intrinsics may vary slightly between different CMS products, due to different hardware being used. These variations have been noted in the text where applicable.



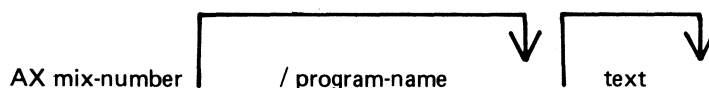
## AX (Accept a message for a program)

This intrinsic allows the operator to communicate with a program in the mix. The program must already be suspended waiting for an "accept" (ACPT).

The MCP will prompt the operator for input by printing "mix number/program-name ACPT" on the SPO.

The maximum length of the "text" or operator input is 50 characters. Operating instructions for individual programs will provide the operator with valid "text" responses.

Format:



Example:

The program BM001 displays a message asking for a file name to be entered. The operator responds with the appropriate text, in this case ARSCHG, by the AX message.

```
BM001
01/BM001 BOJ
ENTER BM202 FILE NAME
01/BM001 ACPT
AX 01 ARSCHG
```

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
mix-number / program-name AX INVALID	Specified program was not waiting for an "accept" or mix number and specified program name do not match.	Check with MX for proper mix-number and program-name combination.

## CL (Clear Peripheral)

This intrinsic allows the operator to clear the peripheral from the program and bring the program to End of Job (EOJ). It breaks the "links" between the program and the peripheral.

For example, if the line printer "hangs" during the printing of a report and an attempt is made to DS the program, it will not be possible to discontinue the program unless the line printer is made ready or CL is used to break the "link" between the program and the line printer.

Format:

CL	{	printer peripheral	}
	{	tape peripheral	}
	{	self-scan peripheral	}
	{	ICMD peripheral	}

Examples:

CL LPA

CL SSA

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
CL peripheral INVALID	Program is not waiting on "hung" peripheral.	Check input.



## DC (Data Communications operator input)

This intrinsic enables the operator to enter messages from the SPO to the Message Control System (MCS) if data communications activity is in process. The message text, after being stripped of the "DC" characters and the following blank character, is transferred to the MCS input message queue and marked as "operator input".

The interpretation of the message text is defined by the particular MCS.

Format:



Example:

To enter the text "MAKE STATION 2 READY":

DC MAKE STATION 2 READY

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
none	MCS input successful	none
DC INVALID	no MCS in the mix	check input; execute the MCS
DC NOSPACE	There is no available message space in memory for this message	wait a short time then re-input message; if unsuccessful several times, request technical assistance.

## DP (Discontinue and Dump)

This intrinsic is similar to the "DS" intrinsic. The difference is that the disk work space (Virtual Memory on Disk, Virtual Disk) is not freed up and returned to an available status.

The disk work space is, instead, updated from memory with all the most current information about the program. The disk backup is then made into a file (locked) and given a name, "DMFILnn" ('nn' is the mix number for user programs, utilities, and MCP intrinsics).

The peripherals and memory in use by the specified program are made available to other programs.

DP is used when a technical analysis of a particular program is required following a failure during its operation.

Format: **DP mix-number/program-name**

Example:

DP 01/GL060

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
mix number/program-name DP'ed	DP successful	none.
input INVALID	mix number does not correspond to program name	Check input (reinput if necessary). or Check with MX for proper mix number and program name combination.
input INVALID - NEEDS PROGRAM-ID	program-name is missing	Check with MX and re-input.

## DS (Discontinue Program)

This intrinsic causes the orderly termination of the specified program. All peripherals in use by the program are made available to other programs.

Format:

DS mix-number / program-name

Example:

To terminate the program AR040 which has mix number 2:

DS 02/AR040

Output messages:

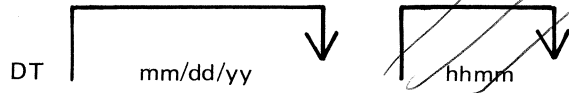
MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
mix/prog DS'ed	DS successful	none
input INVALID	mix number does not correspond to program name; or program is an MCS.	check with MX, reinput
input INVALID-NEEDS PROGRAM ID	program name not specified	check with MX, reinput

Note: if the program is waiting on a "hung" peripheral device, try the CL intrinsic.

## DT (Systems Date and Time)

This intrinsic allows the operator to inquire about or change the system date and time maintained by the MCP.

Format:



Examples:

To inquire about the system date (and time if the system contains a real time clock)

DT

To change the system date:

DT 01/01/78

To change the system date and time:

DT 03/23/78 1234

(March 23, 1978 is the new date. 12:34 is the time).

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
<p>"DD MON YY YYDDD HHMM DOW" where DD = day of month MON = 3 letter abbreviation of month. YY = year YYDDD = Julian date HHMM = time (hours and minutes) DOW = day of week.</p>	<p>Normal response to "DT".</p>	<p>none</p>
<p>&lt;INVALID DATE&gt;</p>	<p>An error was made in one of the follow- ing fields: MM DD YY For example = a MM entry of "0" or greater than 12 is invalid. The entire date is rejected, but a valid time entry in the same mess- age will be accept- ed.</p>	<p>Reinput date portion of message</p>
<p>&lt;INVALID TIME&gt;</p>	<p>A time greater than 2359 was entered.  The time is rejected. A valid date entry in the same message will be accepted.</p>	<p>Reinput time portion of message.</p>
<p>&lt;NO CLOCK&gt;</p>	<p>Time entry was made, but system has no real-time clock.  Valid date entry will be accepted in same message.</p>	<p>none</p>
<p>MM/DD/YY HH:MM</p>	<p>Normal response to DT inquiry (B80C)</p>	<p>none</p>

## FD (Form Define)

This intrinsic allows the operator to define a logical page for a serial printer (SPA) or set top of page for the SPA.

Unless the operator indicates otherwise, the current position is taken as the top of the page.

If the three parameters (HEIGHT, WIDTH, and OFFSET) are specified, then they are used to define a logical page on the SPA. HEIGHT specifies the number of lines on a logical page; WIDTH the maximum number of characters in one line; and OFFSET the number of characters that the printing area is to be offset from the left. An OFFSET of zero specifies the left-most physical position.

WIDTH and OFFSET added together must not be greater than the number of physical print positions on the serial printer. For example, if the physical printer has 255 columns the maximum printing area is given by a WIDTH of 255 and OFFSET of zero. The logical page will remain the same as defined by FD or next warm-start.

Format:

```
FD SPA [ ] height, width, offset ↓
```

Example:

```
FD SPA 66, 120, 5
```

defines a logical page on SPA where height is 66 lines and the printing area is 120 characters wide offset 5 columns from the left (that is, from columns 6 through 125, numbering the left-most column as column 1).

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
FD SPA numbers INVALID	FD specifications for height, width, and/or offset are not acceptable. Attempt to print beyond SPA capabil- ities.	Check input and re- enter.

## GO (Restart a Stopped Program)

This intrinsic allows the operator to restart a program which has been stopped with the "ST" command.

Format:

GO mix-number | / program-name ↓

Examples:

To restart program whose mix-number is 3:

GO 3

To restart program PR020:

GO 3/PR020

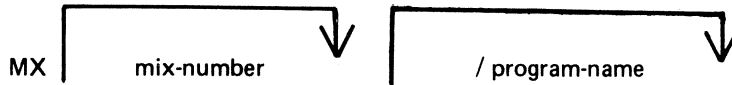
Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
mix-number/prog-name NOT STOPPED	Specified program was not waiting for a "GO" command.	Check with MX for suspended program waiting for "GO" reinput.
mix-number/prog-name INVALID	Optional program name was used and it did not match the mix number specified.	Check with MX for correct mix number and matching program name. Reinput.

## MX (Display Current Mix)

This intrinsic allows the operator to inquire about the status of any program(s) currently processing.

Format:



Examples:

To inquire about all programs currently processing:

MX

To inquire about a particular program:

MX 03/PR020

or MX 03

Output messages:

MESSAGE	PROBABLE CAUSES	SUGGESTED ACTION
INVALID MIX	Specified program is not currently running.	Check input (re-input if necessary).
NULL MIX	No programs are currently processing.	None.
INVALID PROGRAM ID	Optional program name was used and it did not match the mix number specified.	Re-input

For each program specified, the following information is provided:

### MIX NUMBER

a number assigned by MCP to this program as it was loaded into memory.

### PROGRAM NAME

PROGRAM PRIORITY - "A", "B" or "C"

A = lowest priority (that is, application program)

B = medium priority (that is, system utility)

C = highest priority (that is, data communications)



STATUS OF PROGRAM

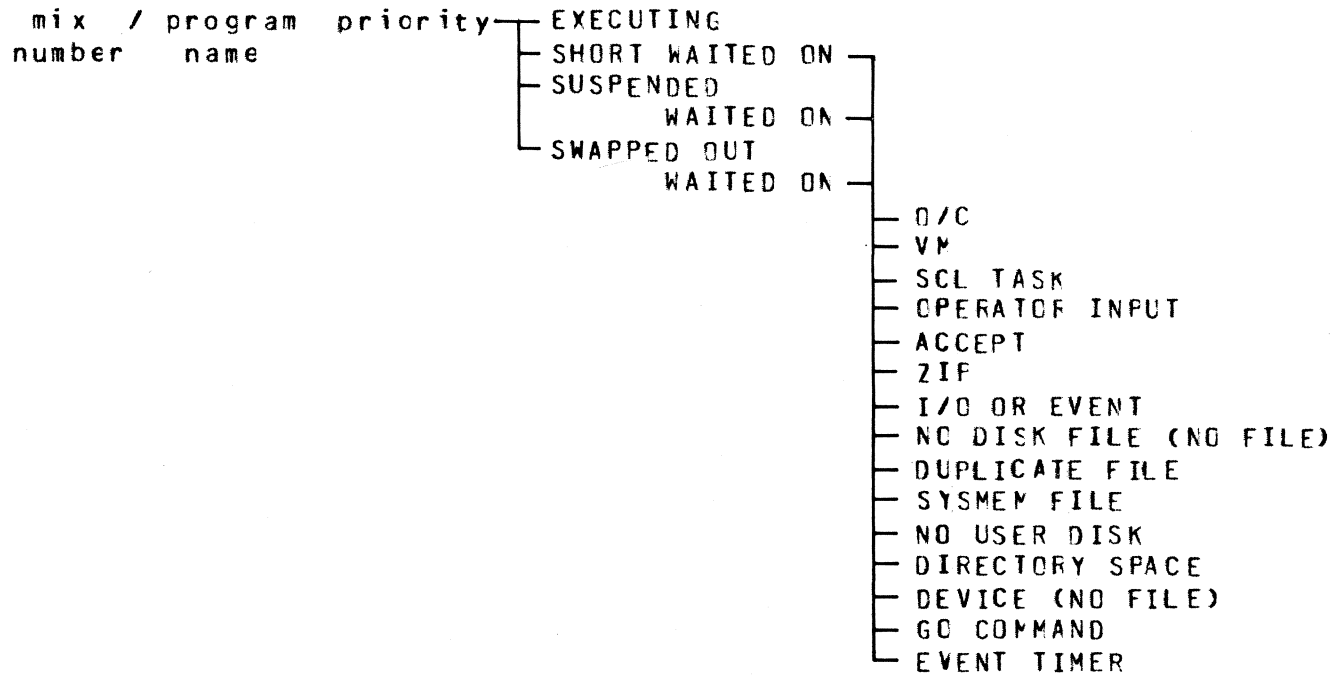
EXECUTING - program processing normally

SUSPENDED WAITED ON - program processing was temporarily halted. For reasons, see chart below.

SHORT WAITED ON - program is waiting on a resource (that is, Virtual Memory or I/O buffer) which the system can guarantee will be made available in a relatively short time.

SWAPPED OUT WAITED ON - portions of this previously suspended program were temporarily removed from real memory and returned to disk. Memory space was required for other programs in the mix. (Reasons for "swap outs" are same as for program suspension).

Possible messages are summarized by the chart below:



Output message examples:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
04/PR060 A EXECUTING	Program processing normally	None
04/PR060 A SUSPENDED WAITED ON O/C	Program is waiting on a file open or close.	None : program will be resumed when file has been opened or closed.
04/PR060 A SUSPENDED WAITED ON VM	Program is waiting on Virtual Memory.	None : do not try to execute too many programs at this time.

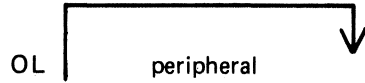
MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
04/PR060 A SUSPENDED WAITED ON SCL TASK	Program is waiting for a "command" from the MCP to be completed (such as response to an "OL" input).	None : program will be resumed when SCL task has completed.
10/LR B SUSPENDED WAITED ON OPERATOR INPUT	Program is waiting for some input from operator. (EX: A program previously suspended by ST requires a GO command to continue).	Provide program with appropriate input. Program will continue processing.
08/GL060 A SWAPPED OUT WAITED ON ACCEP	Program has displayed an "ACPT" message on SPO and is waiting for appropriate response.	Refer to this program's operating instructions for suggested responses to ACPT. Then enter AX, mix number and selected response.
05/AP020 A SUSPENDED WAITED ON ZIP	Program requested assistance of another program in order to complete this job. MCP will automatically load into memory the necessary program(s).	None.
04/PR060 A SHORT WAITED ON I/O	Usually indicative of normal processing, involving I/O activity to disk or peripheral.	None.
05/PR020 A SUSPENDED WAITED ON NO DISK FILE	Program needs (and has not found) a particular file in order to continue processing.	Check SPO for message indicating name of file this program is seeking. Then supply missing file (COPY from backup medium or create it). If in doubt refer to program instructions.
02/PR020 A SUSPENDED WAITED ON DUPLICATE FILE	Program is attempting to place a file of a certain name on disk. However, another file by the same name currently resides on disk. A disk may not contain 2 files with the same name.	Normally, remove the existing file from disk with RM. If in doubt, refer to program instructions.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
10/COPY B SUSPENDED WAITED ON DIRECTORY SPACE	When the disk was initialized the disk directory was constructed to contain a fixed number of file names. The directory has now reached its capacity.	Remove with RM any unnecessary files and program will continue; or DS the suspended program. Replace disk with another disk having sufficient directory space, and re-execute the program.
10/COPY B SUSPENDED WAITED ON NO USER DISK	There is no more available space on disk; or space available is insufficient to hold the file the system is attempting to write; or disk is "checkerboarded".	With KA, analyze amount of available space remaining. If disk is filled remove with RM any unnecessary files; or if disk is filled and a dual-pack file is desired, assign a different disk to this program (see AD intrinsic); or if disk is checkerboarded, use SQ utility to consolidate disk space, then re-execute program that encountered suspension.
10/LIST B SUSPENDED WAITED ON SYSEM FILE	SYSEM file cannot be located.	Request technical assistance.
04/PR060 A SUSPENDED WAITED ON NO FILE B80	Device that a program needs in order to continue processing is either unavailable or not ready; or	RY required device; or assign program to alternative device (see AD intrinsic).
B800	Program needs (and has not found) a particular file in order to continue processing.	Check SPD for message indicating name of file program is seeking. Supply missing file (COPY from backup medium or create).
04/PR060 A SUSPENDED WAITED ON DEVICE B800	Device that a program needs in order to continue processing is either unavailable or not ready.	RY required device; or assign program to alternate device (see AD intrinsic).
02/LF B SUSPENDED WAITED ON GO COMMND	Program was suspended by ST command.	Type "GO" plus mix number of suspended program.

## OL (Request for Status Information of Peripherals)

This intrinsic allows the operator to request the status of peripherals on the system.

Format:



Examples:

To display status of all system peripherals:

OL

To display status of a particular peripheral:

OL DKB

OL LPA

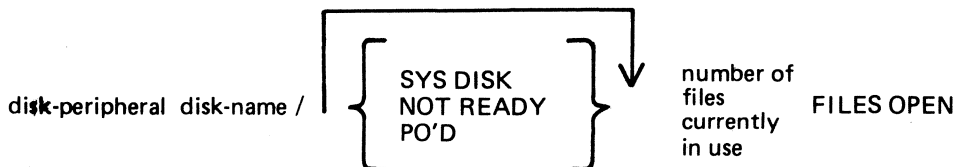
Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
peripheral NOT READY	Peripheral is not on the system; Peripheral may have been "saved"; peripheral may not be correctly loaded.	Check input (reinput if necessary) Ready peripheral if necessary.
OL peripheral NOT CN SYSTEM	peripheral is not configured on machine	none
OL peripheral INVALID	A non-existent device has been specified (that is, OL CCC)	Check input (reinput if necessary).

Other output messages produced by OL depend upon type of peripheral. Refer to the following examples for details.

Examples of disk device output:

The general format of the output message is:



Examples:

DKA AR1/0 FILES OPEN

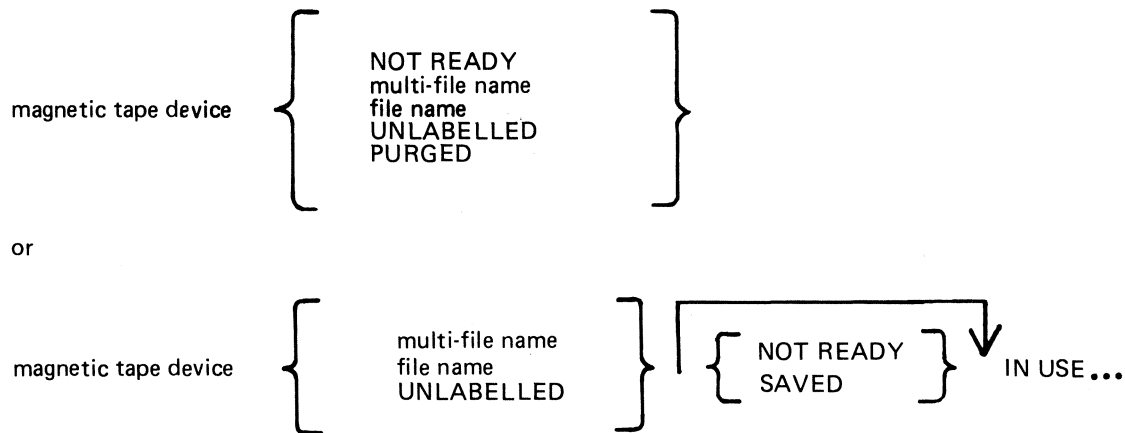
DKB AR2/SYS DISK 2 FILES OPEN

DMA PRA/NOT READY 0 FILES OPEN

DKA AR1/PO'D 0 FILES OPEN

**Examples of magnetic tape device output:**

The general format of the output message is:



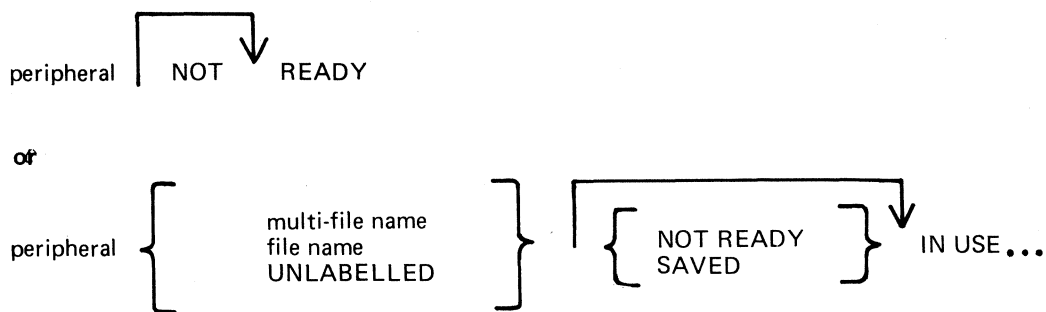
... BY mix-number / program-name

**Examples:**

CTA NOT READY  
 CTA ARTAPE  
 CTA ARTAPE/IN USE BY 10/TAPELR  
 CTA ARTAPE/NOT READY IN USE BY 10/TAPELR

**Examples of output from any other device:**

The general format of the output message is:



... BY mix-number / program-name

**Examples:**

LPA NOT READY  
 LPA NOT READY IN USE BY 04/PR020  
 SSA SAVED  
 SPA SAVED IN USE BY 06/PR060

# TAPE

## PG (Purge Tape)

This intrinsic allows the operator to purge (erase) magnetic tape and cassette tape files, thus labelling them as available for output.

Format:

PG tape or cassette peripheral

Examples:

To purge a cassette tape on drive CTA :  
PG CTA

To purge a magnetic tape on drive MTC :  
PG MTC

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
peripheral * PURGED *	PG successful.	None.
PG INVALID	peripheral not specified in message.	Re-input message
PG peripheral INVALID	Tape could not be purged, as it is "write inhibited", or peripheral is not on the system.	Make certain red tabs on top of cassette are turned inward; make certain "write permit ring" is inserted in tape. Retry PG.

Note: if an attempt is made to purge a tape which is in use, then the response to the OL message for that peripheral is displayed.

## PO (Power Off a disk)

This intrinsic allows the operator to "logically" power off a disk (instruct the MCP that the disk is no longer required). At any time when the MCP is idle it is valid to logically power off the system disk with the PO command. This will cause the MCP to terminate. All systems disk files will be closed and SYS-SUPERUTL will go to End of Job (EOJ).

No disk should be removed from the disk drive, no disk units should be powered down, nor should the main cabinet be switched off, until disks have been logically powered off with PO. Failure to observe this practice might cause disk problems at a later date.

### Format:

PO disk peripheral

### Examples:

PO DKA (disk cartridge)

PO DMB (mini disk)

PO DFA (fixed disk)

### Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
disk peripheral OK or disk peripheral POWERED OFF	Disk was logically powered off.	It is now permissible to physically power off and remove the disk from the disk drive.
disk peripheral REMOVED WITHOUT PO	Disk was physically off before being logically power- ed off.	Check disk for possible corrupted data before re-use.
PO disk peripheral INVALID	specified disk peripheral is non-existent. (ex: PO DKW)	Reinput.
PO disk peripheral NOT ON SYSTEM.	Specified disk is not currently on line.	Reinput.
CANNOT POWER OFF SYSTEM. MIX NOT EMPTY. or PO disk peripheral INVALID	Attempt has been made to PO the system disk while a program is processing.	Allow program to go to End of Job (EOJ), then reinput.

If an attempt to Power Off a disk is made while files on that disk are in use, the OL message for the disk is printed. No further program will be allowed to open files on the disk and when all files in use have been closed, the disk will be logically powered off. If the disk is in use, it will not be powered off immediately after giving the PO command though it will print "disk peripheral PO'ED X FILES OPEN", for all disks.

If a disk is removed without being logically powered off, any program using files on that disk will eventually terminate with an error condition indicating hardware failure.

A PO'd user disk may be made ready again by the RY command or by physically powering the unit off and on.



## PR (Assign Program Priority)

This intrinsic allows the operator to alter the priority of a program by moving it to the highest priority position in the class specified.

Priority "A" is low or normal priority, used for regular work. Within this class, programs which perform more physical I/O operations are given precedence.

Priority "B" is medium priority, used for utilities or programs which may be expected to do emergency work. The priority within this class is reverse historical: that is, a program of this priority placed in the mix will take precedence over previous programs of the same priority.

Priority "C" is high priority, used for data communications programs that are transaction-driven. These are normally dormant, awaiting a transaction, but when required to process a transaction they take high priority to minimize response times. Within this class, programs which do more physical I/O are given precedence.

Format:

PR mix-number / program-name { A  
B  
C }

Example:

To change the priority of mix-number 3 (program REP506) to B:

PR 03/REP506 B

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
mix-number/program-name PR IS priority	Input accepted	None
mix-number/program-name PR INVALID	Mix-number and program-name do not match or priority value incorrect	Check with MX for proper input, and re-enter

## **RY (Ready a Peripheral)**

This intrinsic is used to "ready" a peripheral so the MCP can use it as a resource. When warmstarting, the system will automatically ready all peripherals on the system that are powered on. RY may also be used to Ready a previously PO'd user disk.

Format:

RY peripheral

Examples:

To ready a self-scan:

RY SSA

To ready a line printer:

RY LPA

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
RY peripheral INVALID	Attempt was made to Ready a non-existent peripheral (that is, RY LLP); Attempt was made to ready a device already "ready".	Check input (reinput if necessary)
RY peripheral NOT ON SYSTEM	Attempt was made to ready a peripheral on-line to the computer.	Check input (reinput if necessary).

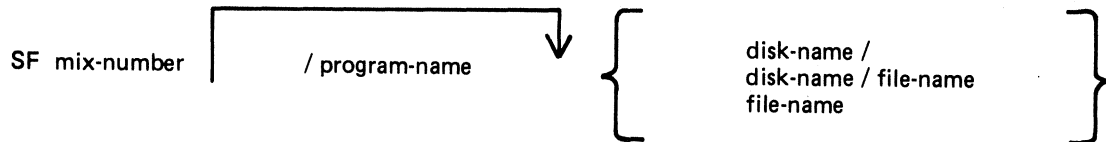
## SF (Substitute Disk File)

This intrinsic allows the operator to direct a program to a particular disk file if it is waiting on a "NO FILE", "NO PACK", "DUPLICATE FILE", or "BAD FILE NAME" condition.

This command causes temporary modification of the program's file parameter block. The modification remains in effect for the current execution only, or until it is remodified by the program during the current execution.

The command can only be used when the program is suspended waiting on one of the above conditions. It is not possible to anticipate the program's requirements and modify the file parameter block in advance.

Format:



Examples:

Program AP10 (mix number 01) requests a disk file called APD2T on disk APD. To direct the program to use file APD2S on the same disk:

```
01/AP10 <10> WAITING APD/APD2T DK NO FILE
SF 01/AP10 APD2S
```

(the first line is the MCP output message; the second is the input SF message in response to the "NO FILE" condition).

To direct the same program to use file APD2T on disk APD1:

```
SF 01/AP10 APD1/
```

or

```
SF 01 APD1/
```

To direct the same program to use file APTEMP on disk ARTD:

```
SF 01 ARTD/APTEMP
```

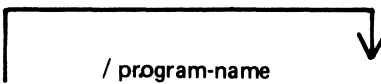
Output message:

MESSAGE	POSSIBLE CAUSE	SUGGESTED ACTION
mix-number / program-name SF INVALID	Program is not waiting on a "no file" or other condition, or mix-number and program-name do not correspond.	Check with MX and re-enter.

## ST (Temporarily Suspend a Running Program)

This intrinsic places a temporary halt on a program that is running. The program still appears in the mix. The data needed to restart the program exactly where it stopped is transferred from memory and stored on disk. The memory that was being used by the "stopped" program is now made available to the MCP for other use. The GO command must be used to restart the program.

Format:

ST mix-number  / program-name

Examples:

To stop the program whose mix-number is 3:

ST 3

To stop the program PR020:

ST 3/PR020

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
mix-number/program name STOPPED	ST successful	none
mix-number/program name INVALID	Program has already been stopped or program is not in the mix.	Check with MX for status of program; (reinput if necessary).

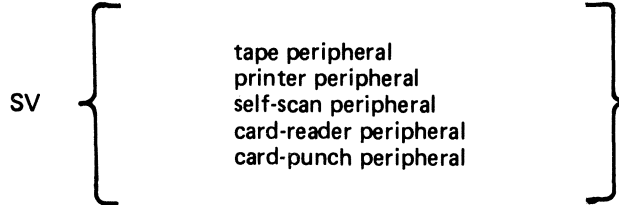
## SV (Save Peripheral)

This intrinsic allows the operator to “logically” power off any input/output device (except disks, see PO intrinsic) in order to prevent their use by any program.

“Tape peripherals” include magnetic tape (MT) and cassette tape (CT).

“Printer peripherals” include line printer (LP) and serial printer (SP).

Format:



Examples

```

SV LPA
SV SSA
  
```

It is possible to “save” a device that is being used by a program. This will allow the program presently assigned to this device to continue using it, but will prevent any subsequent programs from using the device. For example:

```

SV LPA
LPA SAVED IN USE BY 06/PR060
  
```

A “saved” device may be made “ready” again with the RY command or by physically powering the unit off and on.

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
peripheral OK	SV successful	None.
SV peripheral INVALID	Attempt was made to save a disk peripheral or device has already been saved.	Reinput (if necessary using correct peripheral.
SV peripheral NOT ON SYSTEM	Specified peripheral is not on-line to the computer.	Check input; reinput if necessary.
peripheral POWER- ED OFF	SV successful.	None.

## VF (Vertical Format on Printer)

This command allows the operator to define the actions to be taken by the printer when certain vertical format commands are sent. This command applies only to printers which have soft vertical format control.

Format:



The height field specifies the page height in lines. The channel number and line number fields are optional but when specified they must both be present as a pair. The channel number should be 2-11 and page height should not be more than 94.

Example:

```
VF LPA 66, 60, 2 10
```

where page height = 66

end of page = 60

channel number = 2

line number = 10

NOTE: For details see LOAD.VFU utility.

Output Messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
VFU LOAD FAIL - ILLEGAL PARAMETER LIST	Typing error	correct the input and re-enter
VFU LOAD FAIL - peripheral NOT ON LINE	The specified peripheral is not ready	RY the peripheral and re-enter
VFU LOAD FAIL - peripheral IN USE	The specified peripheral is in use by a program	Wait until program has closed the printer file, then re-input
VFU LOAD FAIL - peripheral HAS NO SOFT VFU	The peripheral is not a B9249-30/50 line printer	None

# SECTION 4

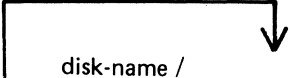
## CMS-COMMON UTILITIES

### INTRODUCTION

This section describes all standard CMS utilities that form part of a CMS system software release. The applicability of any utility depends on the type of hardware available. For example, utilities requiring console files cannot be executed on machines without a console: as an example, CREATE, AMEND and UPDATE cannot be run on a B 1800.

Table 4-1 gives a list of all required peripherals for each utility. In this table, required peripherals are denoted by the letter "R", and optional peripherals by the letter "O". One asterisk ("\*") indicates that out of all the options, at least one is required. In particular, those utilities requiring a line printer may use a console printer by default if the line printer is not present on the system. Two asterisks ("\*\*") indicate that out of all the options, at least two are required.

All the utilities that use initiating message information provide a "star-file" facility. This permits the information to be provided in a disk file instead of from operator SPO input. The entire message, after the name of the utility, may be replaced by an asterisk followed by the disk file name. The format is

\*  file-name

the input could be

RM \* M101A/RMFILE

where RMFILE is a disk file on disk M101A containing one record with the contents

REP200, REP562, RQ=, RCOPY

For all utilities except the compile utility (CO, see section 6), star-files may contain a maximum of five records, and the record-size must be 80 characters. The information should be padded to the right of each record with spaces. No nested star-files are allowed: that is, the information in a star-file may not contain a call on another star-file. If the specified file cannot be found, a "file-name NOT FOUND" message is displayed by the utility.

### SYS-SUPERUTL

This system utility provides the following functions:

- CH - change the name of a file or group of files
- KX - interrogate disk space
- PD - interrogate disk directory
- RM - remove a file or group of files
- IR - initiate recall of SPO log messages
- LB - look back in SPO log
- LF - look forward in SPO log

It will execute automatically if the program file is on the systems disk when one of these functions are required. This program is also automatically executed at warmstart time and **co-ordinates** logging functions at that time.

The utility has some features which can cause the operator confusion. The utility will not appear in the response to the MX command unless it is actually performing one of its functions, when it will appear as 12/PD or 12/CH etc., according to the function which it is currently performing. If an attempt is made to execute one of the SYS-SUPERUTL functions when it is already busy then a response of "<64> LOAD FAILURE UTILITY BUSY" will be returned.

**Table 4-1. Peripherals Required By CMS-Common Utilities**

Utility	con- sole	disk	ser- ial prin- ter	self- scan	line prin- ter	cass- ette or mag. tape	card rea- der	card punch	paper tape	ICMD
ADD		R				R				
AMEND *	R	R	0	0						
CH		R								
CHECKADUMP		R				R				
CHECK.DISK		R								
CO *		R	0		0	0	0			
COMPARE **		0	0		0	0	0		0	
COPY **		0				0	0	0	0	
CP										
CREATE *	R	R	0	0						
DA	R	R	R							
DD		R								
DUMP		R				R				
FL	R	R		R						
FS		R								
ICMD *		R	0		0					R
IR		R								
KA *		R	0		0					
KX		R								
LB		R								
LD		R				R				
LF		R								
LIST **		0	0		0	0	0		0	
LOAD		R				R				
LR *		R	0		0					
MODIFY *	0	R	0		0					
PD		R								
PL *		R	0		0					
RM		R								
SQ		R								
TAPELR *			0		0	R				
TAPEPD						R				
TL		R								
UNLOAD		R				R				
UPDATE *	R	R	0	0						
XD		R								



## LOGGING

3 - SYS-LOG 1  
2  
3

When the system is warmstarted the SYS-SUPERUTL utility will be initiated and SYS-LOG files will be created. The information about the number and size of log-files is stored in a file called "SYSCONFIG" (see CONFIGURER). Then the MCP will initiate a function of SYS-SUPERUTL, which will start up the "TL" utility, and the transfer of log-files to a "SYS-LOG-HOLD" file will begin. When all the log-files are transferred and TL goes to End of Job, SYS-SUPERUTL will remove the old SYS-LOG files and create new SYS-LOG files.

During a session all the console input/output messages that normally appear on the SPO are stored in SYS-LOG files SYS-LOG-01 through SYS-LOG-nn, where "nn" is 03 to 16 (see CONFIGURER). When one log-file is full the messages will be directed to the next log-file. When all the log-files are full the logging will be directed to the first file again. This will overwrite the information held in the SYS-LOG-01 file unless the utility "TL" is begun beforehand, which will transfer all the transferable log-files and keep them in the "TRANSFERRED" state (see TL and WL).

The system will automatically transfer all log-files only at warmstart time.

### COMMON UTILITY OUTPUT MESSAGES:

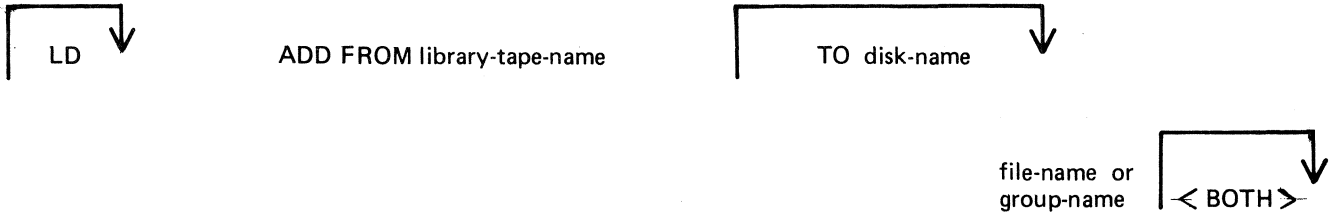
MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
INVALID CHARACTER IN IDENTIFIER	Disk name or file name contains character(s) not permitted by the system. Valid characters are: A-Z, 0-9, . (dot), - (dash).	Check input and re-input if necessary.
ILLEGAL PARAMETER LIST	Typing error.	Check input. After words "ILLEGAL PARAMETERS LIST" system will display portion of input message that contains error.
file-name NOT FOUND or file-name NOT ON LINE	Specified file name is not on disk.	Check input and re-enter if necessary; check for correct disk; supply specified file (COPY file from backup medium or create file).
NO SPECIFICATIONS GIVEN	Input message is incomplete.	Check input and re-enter.
DISK disk-name NOT OPENED NOT ON LINE or DISK disk-name NOT AVAILABLE or DISK disk-name FOR XD NOT AVAILABLE	Specified disk is not on-line to computer.	Check input and re-enter if necessary; Check for correct disk; Ready disk;

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
FILE LIST MAY BE INCOMPLETE	A group of files on an unrestricted pseudo-disk was requested, but the disk is off-line.	
PSEUDO-DISK pseudo-disk-name ON DSK disk-name NOT AVAILABLE	The pseudo-disk specified has not been found.	
DISK disk-name IS A PSEUDO-DISK	The disk specified is not a physical disk but a pseudo-disk: probably the input disk name is incorrect. This message is printed by those utilities which cannot handle pseudo-disks.	Correct the disk name and reinput.

## ADD (Add Files From Library Tape to Disk)

This function, a part of the utility LD, allows the operator to copy files from a library tape to a disk.

Format:



If the <BOTH> option is used immediately after a request to add a keyfile, the data file will also be copied, provided it does not precede the keyfile on the library tape. The keyfile will then refer to the disk ~~which~~ now holds the data file (rather than the disk from which the data file was dumped to the library tape).

A file is copied only if no other files on the specified disk have the same name.

Examples:

To copy all files from ARTAPE to the system disk:

```
ADD FROM ARTAPE=
```

To copy a file called PRFILE from PRTAPE to a disk called PRBU:

```
ADD FROM PRTAPE TO PRBU PRFILE
```

To copy files called GL300 and GL200 from GLTAPE to the system disk:

```
LD ADD FROM GLTAPE GL300 GL200
```

To copy a keyfile called PR240K and its data file from a tape called PRTAPE to a system disk=

```
ADD FROM PRTAPE PR240K <BOTH>
```

Since "ADD" is a part of the utility LD, "LD" is actually what will appear in a mix message. To discontinue the ADD function, "DS mix-number/LD" must be used.

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
file-name LOADED	ADD successful	None.
Library-tape-name NOT A RECOGNIZED DUMP TAPE	Specified tape does not have a valid CMS label, or has not been created by the LD utility (for example: tape is a COPY tape).	Provide correct tape and retry; or DS LD utility.
NO FILES IN THE FAMILY group-name ON TAPE library- tape-name FOR ADD	Specified group was not found on library tape.	Check input and re-input if necessary; Check for correct library-tape.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
NO FILE file-name ON TAPE library-tape- name FOR ADD	Specified file was not found on library tape.	Check input and re-input if necessary; check for correct library tape.
file-name LOAD/ DUMP DISCREPANCY	End of File reached before expected. Disk File Header may be corrupted. Possibly due to mis-reading of tape.	Try tape on different drive in case the drive is at fault.
NO FILES TO LOAD	No files were found on this tape to copy to disk.	Check input and re- input if necessary; Check for correct tape.
file-name NOT LOADED - ALREADY ON DISK. ALTHOUGH WITH DIFFERENT ATTRIB- UTES.	File not copied as a file with the same name already exists on disk. If the 2 files differ in record, block, and file sizes, the "DIFFERENT ATTRIB- UTES" message prints.	Normally remove with RM the duplicate file and re-attempt the ADD.
file-name DATA FILE NOT FOUND ON TAPE FOR LOAD.	Data file for given keyfile does not follow on tape. Data file cannot be copied. Keyfile is copied.	None.

Note: Refer to "Common Utility Output Messages" for additional aid.

## AMEND (Disk File Amending)

This utility is used to modify records within an existing data on source file. The "CREATE" and "UPDATE" utilities use many similar features.

Format:



Input may be either alphanumeric (A) or hexadecimal (N). (See "CREATE" for details). **The default is "A"**

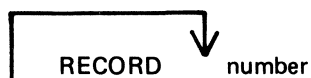
The "number" option may be used to set "tab" positions for character input (see "CREATE" for details).

The utility operates in two modes: "Record Modify" (PK2) and "Record Select" (PK3).

PK1	PK2	PK3	PK4	PK5	PK6
write last & get next	modify	select	---	---	EOJ

PK1 is used to select the next sequential record in the file to be printed. The use of PK1 terminates "Record Modify" and "Record Select" modes, therefore a re-selection of mode must be made before continuing.

If PK3 (Record Select mode) is used, the required record is identified by logical record number using this format:



The "number" may take any value from 1 to the number of records in the file.

PK2 is used to make corrections to existing records. This PK operates as PK2 in CREATE utility (see CREATE for details).

Example:

To amend a source file called MYFILE, record size 40 bytes, tab set at 5, 10, 15, 20:

```
AMEND MYFILE 5 10 15 20
```

First select a record by pressing PK3, and then enter "20" for logical record 20 in MYFILE. Utility selects and prints the contents of record 20.

```
20 ABCDEFGHIJKLMNOPQRST
```

To replace characters, press PK2 and type the replacement

```
D : ZZZZ : OCK1
```

resulting in "20 ABCDZZZZIJKLMNOPQRST"

7

Or if insertion of characters is desired, type the characters to be inserted into the record:

Z : XXXXXX : OCK2

resulting in "20 ABCDZXXXXXXXXZZZOPQRST"

NOTE: the insertion from character six to eleven will result in the shifting of characters "ZZZIJKLMN" from byte position 12 to the boundary of the next tab position, which is 15. Therefore only 3 characters "ZZZ" are shifted from 12 to 14 and "IJKLMN" are lost. The text from the next tab position 15 onwards is not affected.

**Output messages:**

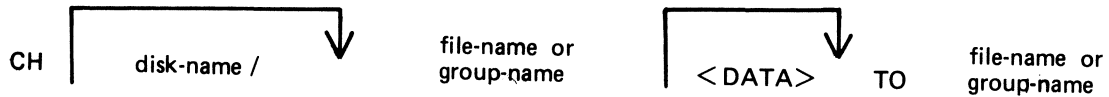
Refer to the section on the "CREATE" utility for output messages.

## CH (Change File Name)

(a function of SYS-SUPERUTL)

This utility allows the operator to change the name of a file or group of files on disk. The <DATA> option allows the data file of an indexed pair to be changed, and it will also cause the keyfile to refer to the new data file name (the keyfile name does not change).

Format:



Examples:

To change the name of a single file:

```

CH BPS320D/DCSTSK36K TO DCSTSK
CH DCSTSK TO INDISK3TSK
  
```

To change a group of files:

```

CH BPS320A/AR= TO BP=
CH PRB= TO PR|=
  
```

To change several different files or groups of files:

```

CH DCSTSK TO INDISK3TSK, BPS320A/AR= TO BP=
  
```

To change the name of the data file of an indexed pair:

```

CH AR200K <DATA> TO AR200BU
  
```

Note: if a change of group file name is specified with the <DATA> option then the data file should appear in the directory after the keyfile. If this is not the case then the name of the data file is changed first, and when the attempt to change the key file name is made, a "data file-name NOT FOUND" message will be displayed. This will not occur when changing the name of a **single** indexed file.

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
"file name" CHANGED TO "file name"	File name successfully changed.	None.
"file name" NOT CHANGED - NOT FOUND	Specified file name is not on disk.	Check input or (re-input if necessary), Check for correct disk.
NO FILES FOUND FOR CHANGING IN THE FAMILY. "group-name"	Specified group name is not on disk.	Check input (re-input if necessary) Check for correct disk.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
"file name" NOT CHANGED IN USE	File's name cannot be changed because it is currently being used by system.	Wait until file is no longer in use, then re-input.
"name" FILE IDENTIFIER TOO LONG	Attempt has been made to change a file name to more than 12 characters in length.	Re-input.
"file name" NOT CHANGED - ILLEGAL REQUEST	Attempt has been made to change the name of a file to "SYSMEM" (a name reserved for system use) or all spaces.	Re-input.
"file name" NOT CHANGED - "file name" ALREADY ON DISK	Attempt has been made to duplicate the name of a file already on disk.	Re-input.
KEYFILE "file name" NOW POINTS TO DATA FILE "file name".	Successful completion of data file name change.	None.
<64> LOAD FAILURE UTILITY BUSY	Another function of SYS-SUPERUTL is being executed.	Wait until SYS-SUPERUTL is free, then re-input.



## CHECKADUMP (Compare Library Tape with Disk)

This utility allows the operator to compare information in files on a library tape with corresponding files on disk. It is used to verify that a library tape is correct after files have been DUMPed or UNLOADed, or that disk files are correct after files have been ADDED or LOAded. Specified tape is processed sequentially, file by file, and the disk is searched for corresponding files. The utility will notify the operator on up to four errors in a given file. If there are more than four errors, it will ignore the rest of that file, and proceed to the next file on tape.

Format:

```
CHECKADUMP library-tape-name WITH disk-name
```

Examples:

To compare files on the tape called PRTAPE with the corresponding files on the system disk:

```
CHECKADUMP PRTAPE
```

To compare files on the tape called ARTAPE with the corresponding files on a disk called ARDISK2:

```
CHECKADUMP ARTAPE WITH ARDISK2
```

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
library-tape-name NOT A RECOGNIZED DUMP TAPE	First record of tape not recognized by CHECKADUMP. Tape may not have been created by LD utility.	None. Utility ends.
library-tape-name INVALID DIRECTORY ON TAPE	More or fewer entries in directory on tape than specified in the first record of tape.	None. (See CMS MCP Reference Manual for additional in- formation on tape formats.
COMPARISON ERROR ON library-tape- name ON DISK FILE HEADERS.	Header in body of tape is not identical to respective header in disk directory. The error count for the file is increased by 1.	Recreate dump tape.
COMPARISON ERROR ON file-name FILE NOT FOUND FOR CHECK.	Corresponding disk file cannot be found for file on tape.	Recreate dump tape.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
COMPARISON ERROR ON file-name FILE NOT AVAILABLE FOR CHECK	Corresponding disk file cannot be read for this file on tape.	Recreate dump tape.
COMPARISON ERROR ON file-name (AROUND RECORD number)	Discrepancy between disk file and tape file. Record number in vicinity of error in file is printed, if possible. One is added to error count for this file.	Recreate dump tape.
COMPARISON ERROR ON file-name AROUND END OF FILE	Difference in length of tape and disk files. One is added to error count for that file.	Recreate dump tape.
COMPARISON ERROR ON file-name DIFFERING FILE SIZES	Difference in sizes of disk and tape files.	Recreate dump tape.
COMPARISON ERROR ON file-name DIFFERING FILE TYPES	Difference in file types of files being compared.	Recreate dump tape.
COMPARISON ERROR ON file-name DIFFERING RECORD SIZES	Difference in record sizes of the files being compared.	Recreate dump tape.
COMPARISON ERROR ON file-name DIFFERING BLOCK SIZES	Difference in block sizes of files being compared.	Recreate dump tape.
NO DISCREPANCIES BETWEEN DUMP TAPE library-tape-name AND DISK disk-name.	CHECKADUMP successful.	None.
DISCREPANCIES FOUND BETWEEN DUMP TAPE library-tape-name AND DISK disk-name.	Discrepancy discovered between disk file and tape file.	Recreate dump tape.

NOTE: Refer to "Common Utility Output Messages" for additional messages.

## CHECK.DISK (Check all Sectors of a Disk)

This utility reads every sector on a specified disk, and reports on any parity errors encountered.

Format:

```
CHECK.DISK [ disk-name ]
```

If no disk is specified, the system **disk** will be checked.

Example:

To read and report any parity errors from sectors of a disk ARBK:

```
CHECK.DISK ARBK
```

If the utility detects that it is being run on 1 megabyte floppy disk, it will give an additional message at end of job if circumstances dictate, as follows:

If 15-30 bad sectors are found then the following message is displayed:

```
DISK disk-name SHOULD BE REINITIALISED SOON
```

If more than 31 bad sectors are found then the following message is displayed:

```
DISK disk-name EXCEEDS BAD SECTOR LIMIT  
PLEASE POWER OFF DISK disk-name
```

If this message is given, then the disk should not be used again.

Output messages:

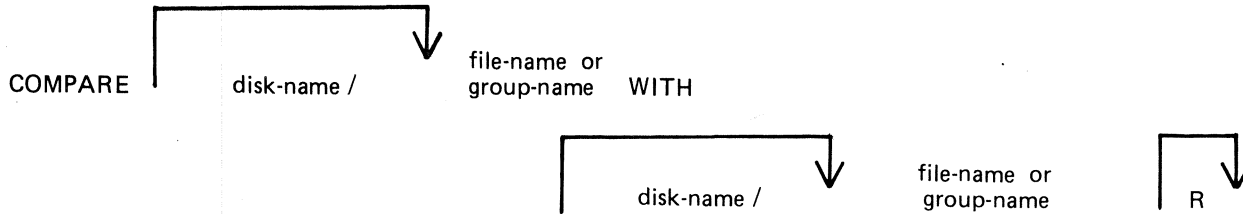
MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
CHECK.DISK ON disk-name COMPLETED - ONE ERROR NOTIFIED.	This message is displayed after all sectors have been read and one error has been detected.	None.
CHECK.DISK ON disk-name COMPLETED - number ERRORS NOTIFIED	The utility has found more than one bad sector.	None.
READ ERROR ON DCL OF disk-name.	An error was encountered while reading sector zero of the specified disk.	None.
ERROR NOTIFIED ON READING SECTOR n	Normal output message. Utility continues.	None.
CHECK.DISK ON disk-name COMPLETED - ERRORS NOTIFIED or CHECK.DISK ON disk-name COMPLETED - NO ERROR	Normal EUJ messages.	None.

Note: refer to "Common Utility Output Messages" for additional messages.

## COMPARE (Compare Files)

This utility compares corresponding records in two files, or in pairs of files within two groups. A realignment feature is also available as an aid to detecting missing records.

Format:



Examples:

To compare file PQ60R on the system disk with file PQ60RS on disk PRB3:

```
COMPARE PQ60R WITH PRB3/PQ60RS
```

To compare the groups of files beginning with AR and the files A27Q on disk ARBK1 and ARBK2:

```
COMPARE ARBK1/AR= WITH ARBK2/AR=,  
ARBK1/A27Q WITH ARBK2/A27Q
```

To compare the file IV20F on the system disk with the file of the same name on disk I32, with realignment:

```
COMPARE IV20F WITH I32/IV20F R
```

If corresponding records are different, the following is printed on a line printer file (or console printer if the line printer is not available).

```
DIFFERENCE DETECTED AT BYTE @nnnn@
```

where n is the number of the byte in the record, starting from 0. The two records are then printed, using more than one line if necessary, with the following format:

```
byte-offset  
32-byte groups in hexadecimal  
32-byte groups in ASCII
```

(A null character (00) in hex is represented by “..”, and a non-printable character in ASCII represented by a blank).

Comparison of groups of files works as in the following example:

Assume DISK1 contains the files A, B, C, D, AB, AC, ABC, BC.

Assume DISK2 contains the files A, B, C, D, AB, AC, ABC, BC, BD, EF.

Then

```
COMPARE DISK1/= WITH DISK2/= compares all files on DISK1 with the corresponding files on DISK2.
```

But

```
COMPARE DISK2/= WITH DISK1/= compares files on DISK2 with the corresponding files on DISK1, and will fail to find DISK1/BD and DISK1/EF.
```

Similarly,

```
COMPARE DISK1/A= WITH DISK2/A= compares files A, AB, AC and ABC on DISK1 with the corresponding files A, AB, AC and ABC on DISK2.
```

Also,

COMPARE DISK1/A= WITH DISK2/AB= compares the following pair of files:  
DISK1/A with DISK2/AB,  
DISK1/AB with DISK2/ABB, (not found)  
DISK1/AC with DISK2/ABC,  
DISK1/ABC with DISK2/ABBC (not found)

The realignment option works in the following manner:

If three consecutive records fail to compare then an attempt is made to compare the third record of the second file with the next two records of the first file.

If all these five comparisons fail then an attempt is made to compare the fifth record of the first file with the fourth, fifth, sixth and seventh records from the second file.

If this comparison fails, then the comparison is terminated with an appropriate message (see later).

If a correct comparison occurs at any stage, then the compared records are used as synchronization for restarting normal comparisons.

For example, consider FILE1 containing 10 records A, B, C, D, E, F, G, H, I and J, and FILE2 containing twelve records K, L, M, N, O, P, Q, R, S, T, U, and V.

The utility compares record A with record K, then B with L, then C with M. If all these comparisons fail, then if realignment is specified record M is compared with records D and E. If this also fails, record E is compared with records N, O, P and Q. If none of these compare, the comparison is terminated.

Note that if there is a missing record in one file, and realignment is NOT specified, a comparison error will arise on every succeeding record until end-of-job.

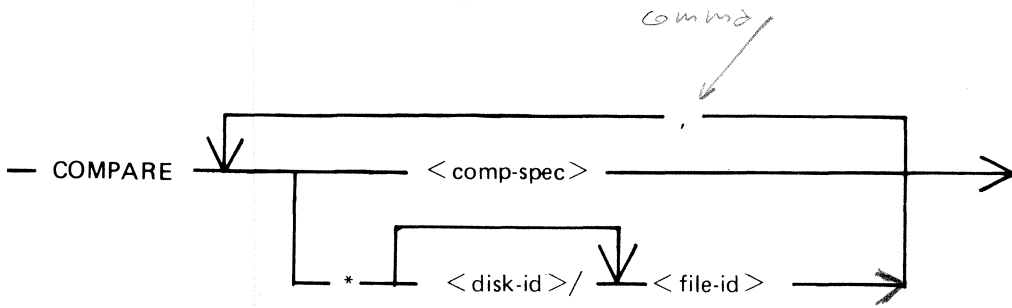
## Additional Capabilities

Further features in this utility are summarized in the railroad chart given in Figure 4-1, which gives the complete input specifications.

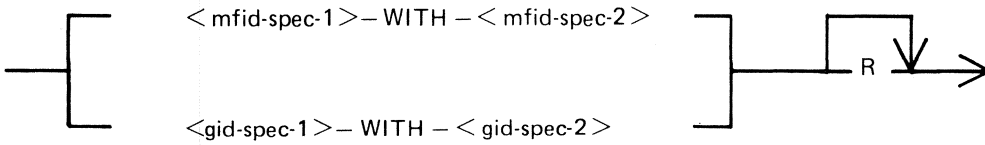
### Non-disk devices:

Files on devices other than disk may be compared by following the file name by one of the following keywords:

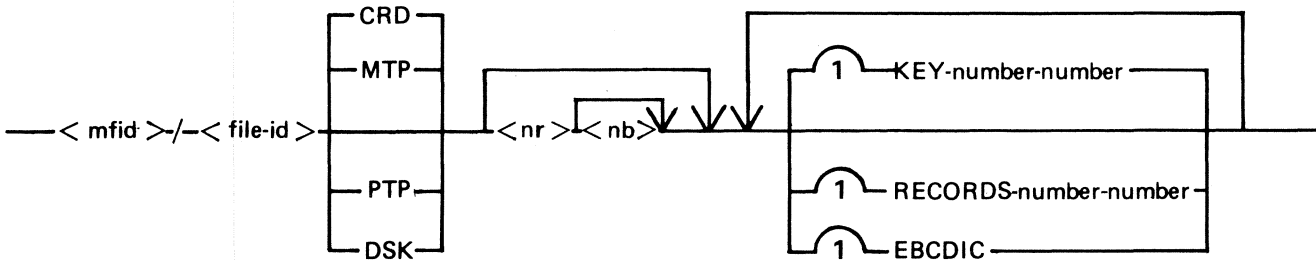
- CRD – any 80-column or 96-column card device
- PTR – any paper tape input device
- MTP – any magnetic tape or cassette device
- DSK – any disk device (the default; this keyword is for documentation only)



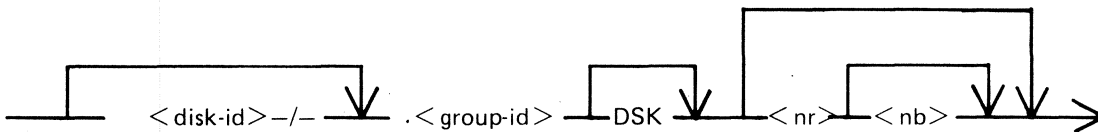
< comp-spec > is defined as :



< mfid-spec > is defined as :



< gid-spec > is defined as :



**Figure 4-1. Railroad Chart for Compare Utility**

**Examples:**

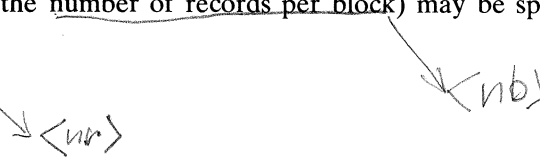
To compare records on a cassette file ARDUMP/FILE020 with a file AR578QQ on the disk WDSK:  
**COMPARE ARDUMP/FILE020 MTP WITH WDSK/AR578QQ**

(note that the two-part name is valid for multi-file tapes or cassettes, refer to section 2 for naming conventions).

To compare two card files DAT1 and DAT2:  
**COMPARE DAT1 CRD WITH DAT2 CRD**

**Record and block sizes:**

The record size (and the number of records per block) may be specified after the file name and device keyword if applicable.



**Examples:**

To compare a system disk file CU265 with a magnetic tape file TPF, treating data blocks on the tape as 80-byte records blocked 9 records to a block:

```
COMPARE CU265 DSK WITH TPF MTP 80 9
```

To compare a system disk file SCR01 containing 90-byte records with a system disk file SCR02 containing 180-byte records, but reblocking the second file as 90-byte records:

```
COMPARE SCR01 WITH SCR02 90 2
```

Note that if the records to be compared are of different lengths, and reblocking is not specified, then only the number of characters in the shorter record are compared.

If EBCDIC is used the file will be translated from EBCDIC on input. The option KEY allows the comparison to be done only on the field defined, the remainder of each record will be ignored. The first number is the offset of the field within the record, the second is its length. If two files have keys of different lengths, the shorter length will be assumed for both the files.

NOTE: The EBCDIC option is applicable when one of the devices is tape

**Examples**

Compare fields starting at byte 11 for 4 characters of FILE1 with FILE2

```
COMPARE FILE1 KEY 10-4 WITH FILE2 KEY 10-4
```

The option RECORDS allows the comparison to be done only on the records specified. The first number is the starting record number and the second number is the total number of records available for comparison. No other record will be read from that file.

**Example:**

Compare records 12, 13, 14 of FILE1 with records 10, 11, 12 of FILE2.

```
COMPARE FILE1 RECORDS 12 3 WITH FILE2 RECORDS 10 3
```

**Limitations:**

*part of records*

The maximum record size is 1024 bytes. If a file exceeds this record size, it may be compared by reblocking. For example, a file with record size of 1200 can be compared by reblocking as 600 bytes blocked 2, or as 300 bytes blocked 4. The higher the blocking factor, the slower will be the comparison. (If the record size is a prime number P, it can be reblocked as 1-byte records blocked P).

The use of a star-file terminates the list of pairs of files to be compared. For example,

```
COMPARE A= WITH DK2/A=, X= WITH DK2/X=,
        STFILE, B= WITH DK2/B=
```

will compare A=, X= and all files mentioned in the file STFILE, but will ignore the comparisons of B=.

**Output messages:**

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
END OF FILE filename BEFORE filename - n ERRORS	End of one file is detected before the end of the other file	None.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
f, lename WITH filename COMPARED - n ERRORS	Normal ending message if both files are same size.	None.
ILLEGAL SYNTAX FOR ITEM input	Initial input mis- typed.	Check input and re-enter.
CANNOT REALIGN filename WITH filename-n ERRORS	Ending message if realignment has been specified but has failed.	None.
INCOMPATIBLE FILESPECS input	A file is specified to be compared with a group of files, or vice versa.	Re-input correct message.
ITEM TOO LONG input	Input message greater than 256 characters.	Divide input into separate parts, and re-enter.
DIFFERENCE DETECTED AT BYTE @nnnn@	See example earlier	See example earlier
CANNOT READ RECORD n OF filename	Parity error on disk file.	Use backup copy of file concerned, if possible.
ILLEGAL KEY FOR file- name: number-number	A key has been specified with a length zero or which does not lie completely within the record. The utility will proceed with next item.	Check the key length and re-input if necessary.
ILLEGAL RECORDS FOR file-name: number	The record number specified for the starting point of comparisons is not present in the file.	None.
CANNOT COMPARE PAST POSITION number IN file-name	The utility limitation came into effect due to a request to compare beyond an offset of number bytes.	None.
file-name EXHAUSTED AT number	The file had a record range specified which ran beyond the end of file. The range has been truncated.	None.



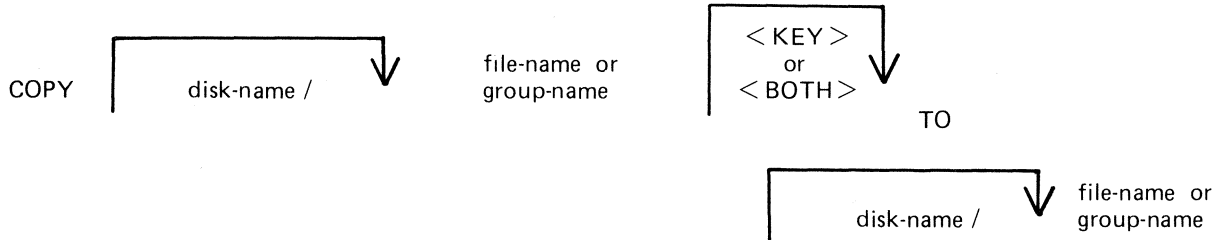
contemporaneamente  
e la dimensione  
record

rispetto alla SAU  
e cancella gli eventuali file che nel disco hanno  
lo stesso nome

## COPY (File Copy)

This utility allows the operator to copy files from one medium to another.

Format:



If as a result of the copying a file a duplicate filename would be created, the original file on the destination disk is removed automatically.

If the file being copied is a keyfile and the <KEY> option is used, the keyfile is copied and the new keyfile refers to the original data file.

If the file being copied is a keyfile and the <BOTH> option is used, the keyfile and the corresponding data file are copied. The data file is given the keyfile name with the letters, "QQ" appended. The new keyfile is made to refer to the new data file name.

If the file being copied is a keyfile and neither <KEY> nor <BOTH> options are used, only the corresponding data file is copied. The records of the new data file are created in keyfile order.

Examples:

To copy a file called AR200 from the system disk to a disk called ARBU:

```
COPY AR200 TO ARBU/AR200
```

To copy files called AR200 and AR300 from the system disk to a disk called ARBU:

```
COPY AR200 TO ARBU/AR200 AR300 TO ARBU/AR300.
```

To copy a file called APTASK from the system disk to APBU, changing its name to APTASKB:

```
COPY APTASK TO APBU/APTASKB.
```

To copy all files beginning with letters "PR" from disk PR2 TO disk called PRBU:

```
COPY PR2/PR= TO PRBU/PR=
```

## Copying Keyfiles

Assume there is a keyfile called PR200K which refers to a data file called PR200.

The statement

```
COPY PR200 <KEY> TO PRB/PR200K
```

 will create a new keyfile PR200K on disk called PRB which references the original data file, PR200, on the system disk.

The statement

```
COPY PR200K <BOTH> TO PRB/PR200K
```

 will create a new keyfile and data file on disk called PRB. The name of the new data file will be PRB/PR200KQQ and the keyfile (PRB/PR200K) will refer to this new data file.

The statement

```
COPY PR200K TO PRB/PR200K
```

 will create a new datafile PR200K on the disk PRB. No new keyfile will be created but the records in the new data file will be created in key order according to the keyfile.

## Additional Capabilities

Further features in this utility are summarized in the railroad chart given in Figure 4-2, which gives the complete input specifications.

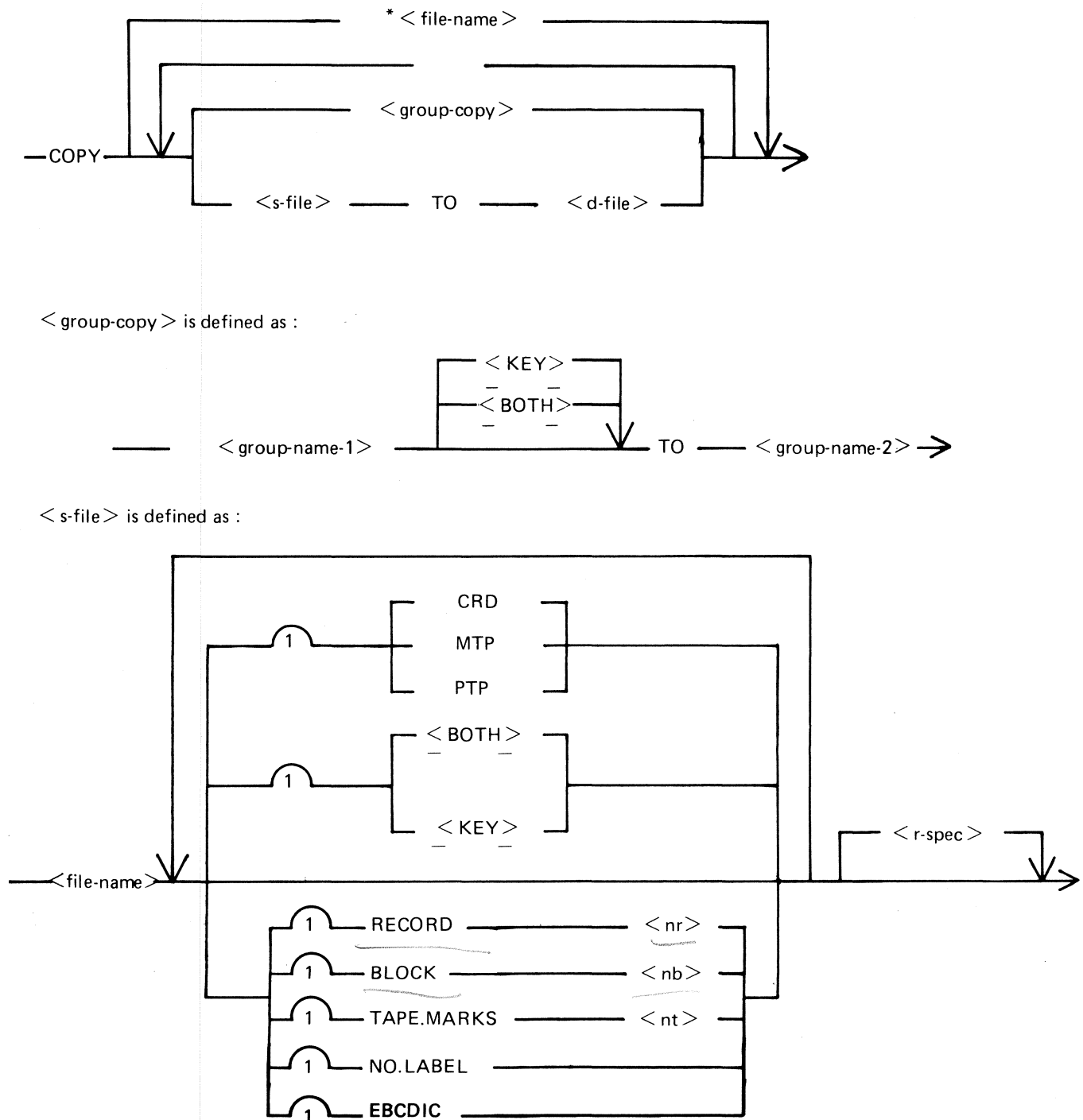
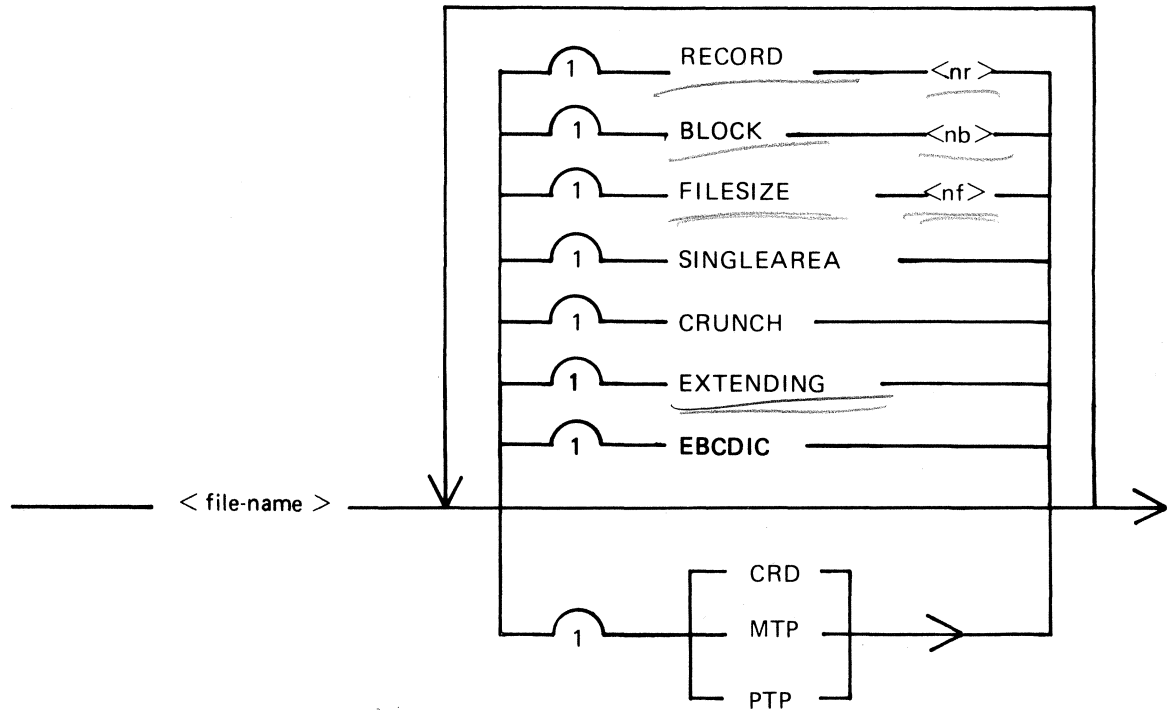


Figure 4-2. Railroad Chart for Copy Utility (Sheet 1 of 2)

< d-file > is defined as :



< r-spec > is defined as :

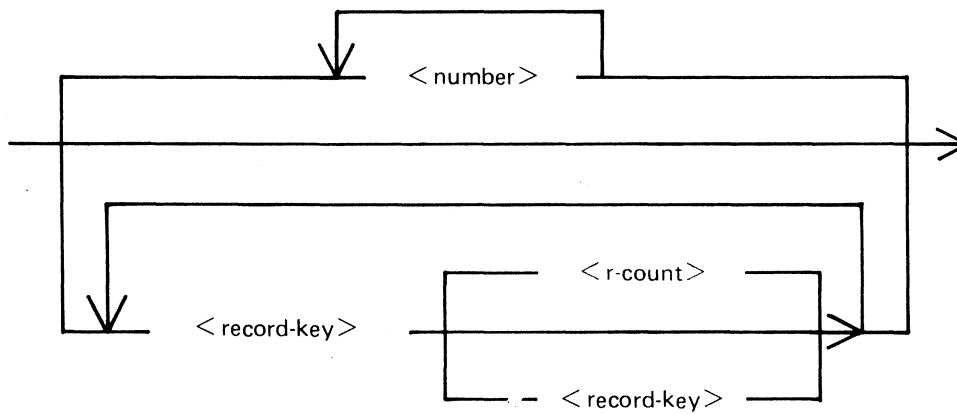


Figure 4-2. Railroad Chart for Copy Utility (Sheet 2 of 2)

## Non-disk devices

Files may be copied to and from media other than disks. Abbreviation for the valid devices are as follows:

MTP – magnetic tape or cassette

CRD – punched card

PTP – paper tape

## Examples:

To copy a cardfile called PRFILE to a disk called PRBU:

```
COPY PRFILE CRD TO PRBU/PRFILE
```

To copy a disk file called PR300 to a single-file magnetic tape:

```
COPY PR300 TO PRTAPE MTP
```

### NOTE

This tape is in “COPY” tape format, not “LOAD/DUMP” format. To access this tape file again it will have to be placed on appropriate device by “COPY” utility, not “LOAD/DUMP”.

To copy a cardfile called PRFILE to paper tape:

```
COPY PRFILE CRD TO PTFILE PTP
```

Note: Paper tapes are always “unlabelled”, and when accessing it, MCP will issue appropriate message requiring an “AD” intrinsic response from operator. See “AD” intrinsic.

## Unlabelled tapes

Input tapes having no CMS labels (“unlabelled” tapes) may be accessed by the COPY utility.

The NO.LABEL option allows the copying of unlabelled files. Upon recognizing an unlabelled file, the MCP will print a “DEVICE REQUIRED” message. The operator must then respond with an appropriate “AD” input message (see “AD”) to identify the unlabelled file.

The end of file recognition for unlabelled files is determined by tapemark count. The TAPE.MARKS option allows the operator to specify the total number of tapemarks which will indicate end of file to the utility when copying an unlabelled file. The default value is 2. Each tape mark which is encountered will contribute to this total. Therefore, a standard labelled CMS file will be copied up to, but excluding, the trailing label if NO.LABEL is specified by itself. (The standard CMS labelled tape format is “label, tape mark, data, tape mark, label”, see CMS MCP manual). The operator must be aware of the format of any file which is to be copied when using the NO.LABEL option.

If the RECORD size is not 180 bytes, refer to the section on Record/Block modifications.

## Example:

To create a disk file called EMPL from first file of a magnetic tape with non-standard label (the format being: LABEL, TAPEMARK, DATA, TAPEMARK):

```
COPY TP MTP NO.LABEL TAPE.MARKS 2 TO EMPL
```

Note: MCP will issue a message asking for unlabelled tape TP. Operator must respond with “AD” input. Additionally, the first record of file EMPL will contain a copy of the non-standard label.

## Record and block sizes

Record and/or Block sizes may be modified for all file types, input and output.

The number of bytes in the record or block is specified using the corresponding “numbers”. The record and block sizes of input disk files are always taken from the file itself (**Disk File Header**). Record and block sizes of non-disk input files are determined as follows:

Record Size:

If RECORD is specified, “number” becomes the new record size.

If RECORD is not specified record size defaults (see below).

Block size:

If BLOCK is specified, “number” becomes the new block size.

If no BLOCK specified, but RECORD is specified, record size becomes new BLOCK size.

If neither BLOCK nor RECORD is specified, Block Size defaults (see below).

Default Values:

Output disk = same as input disk.

Input labelled tape/cassette = from tape label

Input unlabelled tape/cassette = 180 bytes

Cards = 80 or 96 bytes, depending on device.

If the record size is increased then the additional bytes will be filled with spaces if the input file is a source or data file, or with binary zeros for any other type of file.

### Example

To copy an 80-column card file labelled PROGSRC to a disk file called PROGSRC on a user disk “USR”, and make the record size and block size of the disk file 80 bytes and 720 bytes respectively:

```
COPY PROGSRC CRD TO USR/PROGSRC RECORD 80 BLOCK 720.
```

To copy a disk file PRBU/PR300 to magnetic tape **with** large blocks suitable for tape media:

```
COPY PRBU/PR300 TO PRTAPE MTP RECORD 180 BLOCK 1800
```

### File size

The “FILESIZE attribute” of a disk file may be specified for the output disk file. Note that only assigned areas are copied. This feature does not increase disk space at the time of copying, but allows programs to add further records if required. At that time extra disk space may be needed.

### Example:

To copy FILE1 and increase its “FILESIZE” to 1500, replacing the original by the copy:

```
COPY FILE1 TO FILE1 FILESIZE 1500
```

### Single area

The “SINGLEAREA attribute” may be specified for the output disk file. This ensures that the new file will occupy a single disk area.

### Example

```
COPY FFLE2 TO FILE2 SINGLEAREA
```

## Crunching files

The "CRUNCH attribute" may be specified for the output file. This causes any unused disk space at the end of the file to be returned to the system.

### Example:

```
COPY PRB/PR200 TO PRB78/PR200 CRUNCH
```

#### WARNING

A file cannot be "uncrunched" once it is crunched. This means it cannot be extended. It can only be used for inquiry. This option is therefore useful for storing history files.

## Extending disk files

Records can be added to the end of an existing disk file with the option "EXTENDING". The existing file must have identical attributes to the file being copied.

### Example:

A data file called DFTUES was created with Tuesday's data. To add this data to the end of a file called DFMON (containing Monday's data):

```
COPY DFTUES TO DFMON EXTENDING
```

(Note the size of DFMON must be large enough to contain all required records.)

## Selected file copy

Selected record numbers from the input file may be copied.

### Example:

To copy 500 records starting at record #1200 from file FILE1 to file FILE2:

```
COPY FILE1 1200 500 TO FILE2
```

### Note:

pairs of numbers may be specified within each pair; the first number specifies a relative record number and the second specifies number of records to be copied. If an extra number is specified, the last number specifies copying from that record to the end of the file.

### Example:

To copy records 100 to 149, 300 to 499, and 1000 to end of file:

```
COPY FILE1 100 50 300 200 1000 TO FILE2
```

## Selected index file copy

For indexed files, copying of records can be selected based on content of the key. There are 2 options: the number of records can be specified, or an ending key value.

### Examples:

PQR is a keyfile containing personnel records. To copy 15 records from the corresponding data file starting from the record with personnel #01786 to a data file, PSNL:

```
COPY PQR 01786 15 TO PSNL
```

Using same keyfile, to copy all data records from personnel #01786 to 18000 to data file, PSQL:  
 COPY PQR 01786- 18000 TO PSQL

Note:

The second option is specified by the hyphen in the COPY statement. Note that at least one space is required before and after all key values (personnel # in this case).

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
number, number IN file name NOT COPIED	Error found in pair of numbers for selected record copy option. One record number in the pair indicates a section of the file address than a previously specified section. This pair is ignored by COPY.	Check input and re-enter.
file-name EXHAUSTED	End-of-file was encountered while the section of the file indicated was being copied.	None
filename TO filename BAD ATTRIBUTES	Particular attribute specifications is meaningless or inconsistent. The inconsistencies will be in relationship between output device, record size, and block size; or attempt was made to add records to a file whose attributes differ from those of input file.	Check input and re-enter.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
FILE IDENTIFIER file-name TOO LONG	Copy of a group of files was requested. At least one file-name that was to be generated by the copy would have been more than 12 characters long, maximum length of file-names is 12 characters.	Check input and re-enter.
filename TO filename COPY DISCREPANCY	End-of-file reached before expected. Disk File Header may be corrupt.	
filename TO filename COPIED	COPY successful.	None.
filename TO filename ILLEGAL REQUEST	Copy of a group of files was requested. An individual file-name of "SYSTEM" or "(spaces)" was to have been produced. "SYSTEM" (a file-name reserved for system use only) and "(spaces)" are not acceptable file-names.	Check input and re-enter.
filename NOT ACCEPT- ABLE RECORD SIZE OF m EXCEEDS MAXIMUM FOR THIS RUN - RESUBMIT	Copy has encountered a file with a record size greater than expected. This can happen if a magnetic tape file with a record size greater than 1024 characters is submitted to the utility without the record size being properly specified in the initial input.	Check input and re-enter.
filename EXHAUSTED DURING RANGE record- key number number	End-of-file encountered while section of file indicated by "record-key" "number" is being copied.	None



MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
filename EXHAUSTED DURING RANGE "record-key" "record-key"	No records were found in the range "record-key" "record-key".	Check input and re-enter.
NO RECORDS FOR COPYING FROM filename	Specified file contains no records for copying.	Check for correct filename.
INPUT RECORD number OUTPUT RECORD number PERMANENT ERROR ON INPUT FILE	Error encountered- utility cannot continue.	Do selective copies of the parts of the file before and after the bad record, as part recovery
INPUT RECORD number OUTPUT RECORD number PERMANENT ERROR ON OUTPUT FILE	Error encountered- utility cannot continue.	Remove the disk area in error with XD utility and rerun the COPY
INPUT RECORD number OUTPUT RECORD number OUTPUT FILE TOO SMALL	Error encountered- utility cannot continue.	Re-input the COPY with suitable FILESIZE option.
filename TO file- name COPY FAILURE	COPY unsuccessful irr- recoverable error was encountered. The reason should be displayed before this message.	None.
NO RECORDS IN THE KEYFILE - file-name	<BOTH> option was used. Utility was not able to access to a data file through some failure in the keyfile.	None.
file-name NOT FOUND FOR EXTENDING	Specified file was not found.	Check input and re-enter if necessary; Check for correct medium;
filename REMOVED	COPY successful. If any duplicate files are encountered by COPY they are aut- omatically removed.	None.
filename TO filename SELECTION CRITERIA IGNORED	Confirms that select- ion criteria were specified when copy- ing a pair of files, and have been discar- ed.	

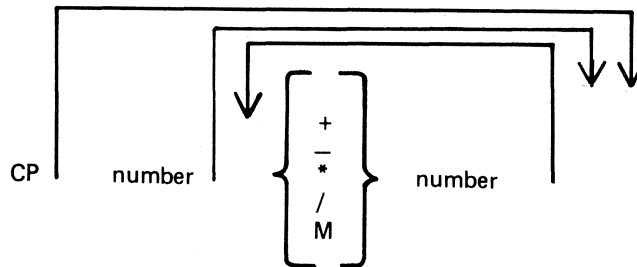
MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
BAD ATTRIBUTES SPECIFIED	Inconsistent attributes were specified for input file.	Check input and re-enter.
filename TO filename EXTENDING FLAG IGNORED	EXTENDING option was ignored by utility when pair of files was copied.	

## CP (Compute)

This utility allows simple computations to be made, with the answer displayed in decimal and hexadecimal. Input may be either decimal or hexadecimal. Hex values must be enclosed in @ symbols.

The utility may be initiated with a single computation to perform, in which case it will do the calculation and terminate. If no calculation is initially provided, the utility issues an ACCEPT to enable the computation to be entered. In this case the utility will do the calculation and then issue further ACCEPTS until a null input to the ACCEPT is given.

Format:



The numbers accepted are any decimal or hex values in the range  
0 9999999999999999[38D7EA4C67FFF@]

or the negative equivalent.

Parentheses are not allowed. The calculation is performed on a strictly left-to-right basis. The operators +, -, \*, /, and M are for addition, subtraction, multiplication, division and modulus division (the result is the remainder) respectively.

Examples:

To compute the hexadecimal value of the decimal number 12345:

CP 12345

To compute the value of a complex expression:

CP 555 \* 3 + 2-100 \* 2/5

(Note: the result of the above is 626, because the calculation is done strictly left-to-right).

Output messages:

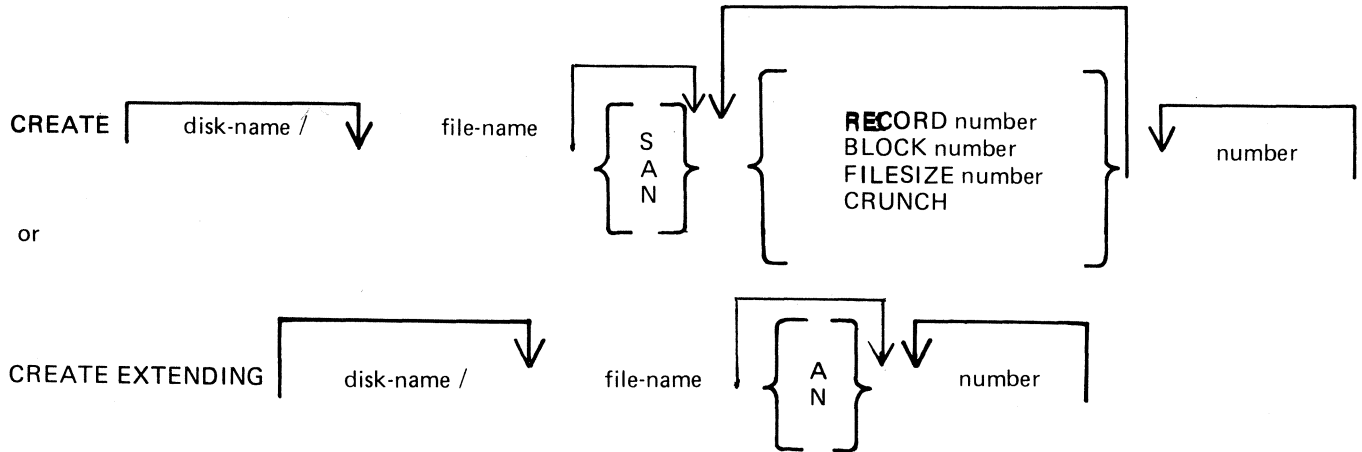
MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
CP : NUMBER TOO LARGE	At least one number is out of range.	Re-input.
CP : MISSING OPERAND	Two consecutive operators (+, etc) have been entered with no number between.	Re-input.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
CP : HEX. NO. WITH MISSING "a"	Illegal symbol; or missing a symbol at front or back of a hex value.	Re-input.
CP : INVALID OPERATOR	Illegal symbol; or an operator (+, etc) has been omitted.	Re-input.
CP : OVERFLOW	An intermediate value in the computation is greater than the maximum value.	Re-input.
CP : DIVISION BY ZERO	Self explanatory.	Re-input.

# CREATE (Create Disk File)

This utility allows the operator to create or extend data or source disk files. The "AMEND" and "UPDATE" utilities use many similar features.

Format:



When creating a new disk file, certain attributes may be specified.

If the S option is selected, a source file will be created using alphanumeric input. If the A option is used a data file will be created using alphanumeric input. The N option creates a data file with hexadecimal (numeric) input. If none of these is selected, S is assumed. Alphanumeric input is accepted as typed, but numeric (hexadecimal) input requires two characters (0-9, A-F) for each byte of the record.

The RECORD option allows the operator to specify the number of characters per record of the new file. If no record size is specified, a record size of 80 bytes is assumed for source files, and 180 bytes for data files.

The BLOCK option allows the number of characters per block of a new file to be defined. The defaults are as follows:

If RECORD size was specified but no BLOCK, BLOCK size will equal RECORD size.

If neither RECORD nor BLOCK is specified, RECORD size will be 80 bytes for source files and 180 bytes for data files; BLOCK size will be 160 bytes for source files, 180 bytes for data files.

The FILESIZE option allows the maximum number of records likely to be written to the new file. This is useful in allocating only as much disk space as required by the file. Once the FILESIZE has been specified for a file, that file can never be extended beyond that number of records. However, the COPY utility may be used for increasing the FILESIZE of an existing file. The default is 2,048 records.

The CRUNCH option allows the operator to specify that the new file should occupy the minimum area of disk, but never be extended.

The numbers specified for the "numbers" option may be used to set "tab" positions within the record (similar to setting "tabs" on a typewriter). If tabs are set, the operator may input data, press OCK1, and the utility will reposition the print mechanism to the next tab position within the record, and await data input. During this repositioning CREATE will fill all character positions left unspecified in the record with a "filler" (ASCII space for source input, ASCII zero for alphanumeric input, and binary zero for numeric zero). The record length plus one will be used as a termination tab position, whether or not other tab positions are specified.

CREATE can be used for record sizes up to 500 bytes, but since the utility cannot be given input greater than the width of the console, tab positions are mandatory on files of larger record sizes. For example, a file of 180 byte records requiring alphanumeric input will require at least one tab position (for instance at position 100). A file of 180 byte records requiring hexadecimal input will require a minimum of three tab positions (for instance at positions 50, 100 and 150). The maximum tab size is 111 in alphanumeric input and 54 in hexadecimal input. That is, the difference between two consecutive tab positions should be less than or equal to 111 in alphanumeric input and less than or equal to 54 in hexadecimal input.

The **EXTENDING** option is used to add records to an existing file. The attributes, such as RECORD and BLOCK sizes, are taken from the old file. The file type is also taken from the existing file. The operator may specify "A" alphanumeric input or "N" for hexadecimal input. If neither "A" nor "N" is specified, "A" is assumed.

**Examples:**

To create a source file called "ICFILE", record size 100 bytes with 5 records per block, tab position set at 65:

```
CREATE ICFILE RECORD 100 BLOCK 500 65.
```

To create a source file called "ICFILE" with Record Size 80, block 3, and a maximum of 20 records in the file:

```
CREATE ICFILE RECORD 80 BLOCK 240 FILESIZE 20.
```

To extend a source file called "ICFILE" (note: the utility will automatically prompt the operator for the next sequential record number to be created):

```
CREATE EXTENDING ICFILE
```

To create a data file called "CFILE" for hexadecimal input with tab positions set at 50, 100 and 150. (Note: Default record size is 180, block 1):

```
CREATE CFILE N 50 100 150
```

The utility operates in two modes: "RECORD INPUT" (entered through PK1) and "RECORD MODIFY" (entered through PK2).

PK1	PK2	PK3	PK4	PK5	PK6
input	modify	—	—	—	EOJ.

*non occorre PK1 perché il programma è cambiato*

The "Record Input Mode" (PK1) is used to enter new records through the keyboard. Characters are input followed by OCK1 for each tab position.

The "Record Modify Mode" (PK2) is used to make corrections to the last record input. The point in the record at which alterations are to be made is selected by typing an identify group of characters immediately preceding the byte(s) of the record to be altered. The portion of the record to be replaced or inserted follows the identifying characters, delimited by a colon (:). If alterations are to be made at the beginning of the record, no identifying characters are necessary.

If OCK1 is used to terminate input, the characters to be altered will replace the corresponding number of characters in the record.

For example, for a record containing "ABCDEF", the amendment C:XY:OCK1 will result in "ABCXYF".

If OCK2 is used to terminate input, the characters delimited by colons (:) will be inserted at the indicated point. The insertion can cause characters in the record to be moved to the right. The shifting of characters applies only to those characters from the starting byte to the next higher relevant tab position; characters beyond this tab position will not be affected.

For example, a record specified with tab positions at 4 and 8, contains "ABCDEFGHJIJ". The amendment C:WXY: OCK2 will result in "ABCWXYDHIJ".

Initially the utility will be in the "Record Input Mode", and on completion of an entry in any mode, it will allow the operator to select the mode not in use, or terminate the utility (with PK6). Unless otherwise instructed, it will continue in the existing mode.

If the FILESIZE is specified and records are entered beyond the given filesize, then the error message is displayed after (filesize + 2) records have been entered. The last two records will not be written, due to the blocking of the output file by CREATE. For example, if the request

```
CREATE ICFILE FILESIZE 15
```

is followed by more than 15 records entered, then the message 'OUTPUT FILE TOO SMALL' will be given after the seventeenth record is entered; and the utility will go to EOJ without writing records 16 and 17 to the output file.

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
FILETYPE IS NOT SOURCE OR DATA	A file other than a "source" file or a "data" file was specified.	Check for correct file; check input and re-enter if necessary.
ILLEGAL PARAMETER LIST - BAD ATTRIBUTE	Specified record and block sizes are incompatible.	Check input and re-enter;
ILLEGAL PARAMETER LIST - TABS ERROR	Tab positions beyond end of record were specified; or input fields larger than capabilities of console.	Check input and re-enter.
NOT HEXADECIMAL CHARACTER INPUT - RESUBMIT	Character other than 0-9 and A-F was input.	Check input and re-enter.
ODD NUMBER OF HEXADECIMAL CHARACTERS INPUT	Warning message indicating that an odd number of hexadecimal characters was input. When input mode is "hexadecimal", utility expects even number of input characters.	None. Utility accepts the input, but adds a zero onto the end (right) of the input, to even it out.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
INPUT ERROR - RESUBMIT RECORD MODIFICATION	Input error during "Record Modify Mode".	Check input and re-enter record modification.
BYTE WITHIN RECORD SPECIFIED NOT FOUND	The identifying string of charact- ers for record modification could not be found in the record specified.	Check input and re-enter the record modification.
UNWANTED KEY PRESSED - PLEASE RE-INPUT.	Invalid OCK was pressed.	Re-enter input and terminate the entry with the correct OCK.
INPUT IMMEDIATELY BEFORE PK6 HAS BEEN LOST	Characters were input immediately before PK6 was pressed to terminate the utility. These characters will not be written to the specified file.	Restart the utility using modify mode to correct this record if desired.
OUTPUT FILE TOO SMALL	Attempt was made to add records to specified file beyond its maximum filesize. File is closed with lock.	List the file to check which records have been entered: then use CREATE EXTENDING to add the desired records. Alternatively use UPDATE with the FILESIZE option.
PERMANENT ERROR ON OUTPUT FILE	Utility is not able to write any more records to the new disk file because of errors on the disk. File is closed with lock.	List the file to check which records have been entered: then COPY to a different file and continue using CREATE EXTENDING.
RECORD SELECTION ERROR	Attempt was made to select a non-exist- ant record.	Check input and re-enter if necessary.
ILLEGAL RECORD NUMBER SPECIFIED	Attempt was made to select record greater than the number of records in file, or zero.	Check input and re-enter if necessary.



MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
RECORD REQUESTED IS BEYOND E.O.F.	Attempt was made to select record beyond end of file.	Usually indicates update is complete: utility terminates normally, no action need be taken.
E.O.F. REACHED DURING DELETIONS	Attempt was made to delete a record beyond end of file.	Usually indicates file update is complete: utility terminates normally, no action need be taker.
FILE TYPE IS NOT DATA	Input type A or N has been specified and the type of file given for extension is not data.	Check the file type and re-input correct file type.

Note: Refer to "Common Utility Output messages" for additional messages

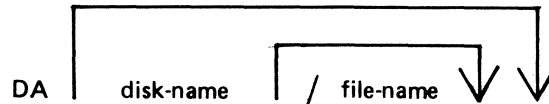
N5

## DA (Disk Analysis)

This utility allows the operator to read any portion of a CMS-format disk. It is an interactive program, with the operator entering a range of commands via the console.

The utility is commonly used to print the contents of the disk directory. In general, if the PD utility operates correctly for a specific disk, then DA should also run successfully. Specifically, the disk cartridge label, the name list entry and disk file header for SYSMEM must be intact. (Refer to the CMS MCP manual for details of the disk format and directory structures).

Format:



The utility operates in two modes “disk mode” and “file mode”. If no file name is specified the utility operates in “disk mode”. If no disk-name is specified, the system disk is analyzed.

## Disk Mode

In this mode the operator can enter via the keyboard a number of commands. These commands can be abbreviated according to the table provided at the end of this section. The format and meaning of each command in disk mode is given below.

END

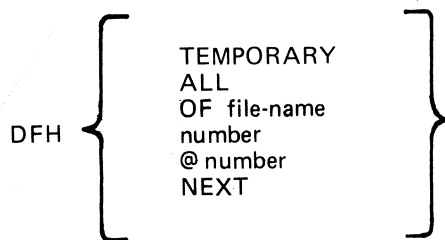
Terminates the utility.

DCL

Reads and formats the contents of the disk cartridge label.

DFH

Reads and formats the contents of selected disk file headers. This command is followed by other details, given here:



The “TEMPORARY” option displays the headers of all temporary files.

The “ALL” option displays the headers of all files, and their contents if in use.

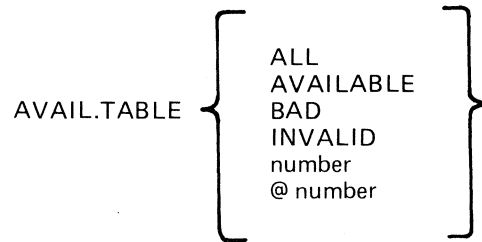
The “OF” option displays the header of the specified file: all headers will be checked and duplicates will be displayed if found.

The “sector-number” option displays any sector in disk file header format, where the number is a decimal sector address. If preceded by an @ symbol, the sector-number is in hexadecimal. This feature can be used after displaying other parts of the directory, which include sector addresses for disk file headers in hexadecimal.

The “NEXT” option displays the header of the next file in the directory.

## AVAIL.TABLE

Reads and formats the contents of selected parts of the disk available space table. This command is followed by other details, given here:



The “ALL” option displays the entire available table.

The “AVAILABLE” option displays entries for available area only.

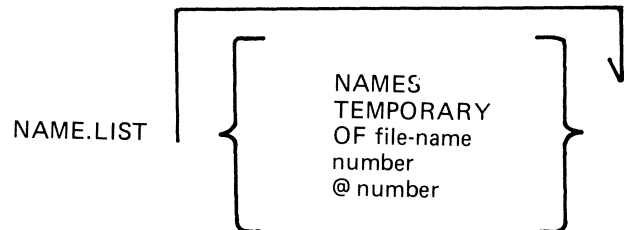
The “BAD” option displays entries for bad sectors only.

The “INVALID” option displays all entries in the available table which are invalid, in that the “length” entry does not equal the difference between “start address” and “end address”.

The “sector-number” option displays any sector in available-table format, where the number is a decimal sector address is in hexadecimal.

## NAME.LIST

Reads and formats the directory name list, including the sector addresses of associated disk file headers. This command may be followed by other details, given here:



If no further details are given, then the entire name list is displayed.

The “NAMES” option displays entries for old (existing) files only.

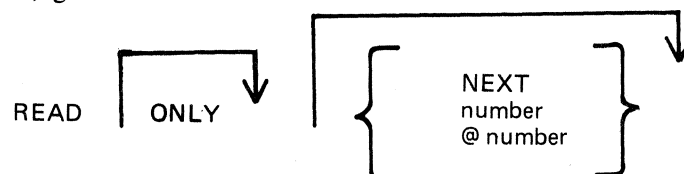
The “TEMPORARY” option displays entries for temporary files only.

The “OF” option displays the entry for the specified file : all entries will be checked and duplicates will be displayed if found.

The “sector-number” option displays any sector in name-list format, where the number is a decimal sector address. If preceded by an @ symbol, the sector-number is in hexadecimal.

## READ

Reads and displays the contents of any specified sector in hexadecimal and ASCII format. This command may be followed by other details, given here:



The "ONLY" option inhibits the display of the information.

The "NEXT" option will read the next sector. Note that after some operations which involve a search, the "next" sector may be indeterminate. After a READ of sector n, a READ NEXT will read sector n+1. A READ command with no further details is taken as a READ NEXT.

The "number" option reads the sector whose address is the number. If preceded by an - symbol, the sector-number is in hexadecimal.

## DISPLAY

Displays the current sector address and contents of the program's sector-buffer. The first DISPLAY command must be preceded by a READ command. A READ ONLY followed by a DISPLAY is equivalent to a normal READ.

## KFPB

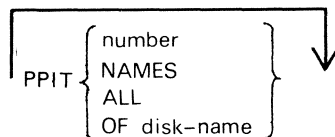
The Key File Parameter Block of a keyfile can be printed.

KFPB OF file-name

If the file specified is a key file, then record 1 of that file is selected and the information is printed.

## PPIT

The Pseudo Pack Identification Table can be found from the Disk Cartridge Label and requested information may be printed:



The "number" option displays the sectors from the PPIT.

The "NAMES" option displays the used entries of the PPIT.

The "ALL" option displays every sector of the PPIT.

The "OF disk-name" option displays the entry for that disk only.

## File Mode

In file mode, the utility can be used to read selected records. Only the following commands are valid:

READ  
DISPLAY  
END

The READ command has the same format as in disk mode, except that the "number" refers to the logical record number in the file, and a READ NEXT will read the next logical record in the file. The amount of information displayed is equal to the file's record length. In DISPLAY option, if the file is a key file, then the KFPB of that key file is displayed.

## General Notes

In disk mode the sector number starts from zero; that is, "READ 0" will read the first disk sector.

In file mode the record number starts from one; that is, "READ 1" will read the first logical record.

Any I/O error causes the "fetch value" to be displayed, with the current sector address if in disk mode, or current record number if in file mode.

## Abbreviations

For ease of use, input commands and other keywords may be abbreviated, as in the following table:

READ	R
ONLY	G
NEXT	N
DISPLAY	D
END	E
DCL	DC
DFH	DF
ALL	AL
OF	OF
TEMPORARY	T
AVAIL.TABLE	A
AVAILABLE	A
BAD	B
INVALID	I
NAME.LIST	N
NAMES	NA

### Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
OUTWITH format AREA ADDRESS: number LENGTH number	In disk mode the sector which has been specified to be read in DFH, NAME LIST, AVAILABLE TABLE or PPIT format is outwith relevant area.	None.
FILE IS NOT A KEYFILE	The KFPB option has been requested and the file which was specified is not a keyfile.	None.
NOT FIXED DISK SYSTEM	The PPIT option has been requested on a non-Fixed Disk system and no Pseudo Pack Identification Table exists.	None.
DA TERMINATED	The END command is successful.	None.
ILLEGAL PARAMETER input	No valid disk name of file name has been specified in the initiating message.	Re-enter

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
SUCCESSFUL READ REQUIRED FIRST	A DISPLAY command has been entered before a success- ful read.	Enter a READ command, then a DISPLAY will work.
NOT VALID IN FILE MODE	A command other than READ or DISPLAY has been used in file mode.	Check input.
NONE FOUND or NOT FOUND	No information has been found in answer to a command.	None.
READ ERROR ON DCL OF disk-name	An error was encountered while reading sector zero of the specified disk.	None.
PARAMETER REQUIRED	The option which has been specified is incomplete.	Re-input the complete option.
LAST OPTION UNSUCCESSFUL	The DISPLAY option has been requested before a sector or record has been read or after an option has been unsuccessful.	Perform a read before using display option.
ERROR ON READ- RECORD:----	The disk parity error has been encountered while reading a record in file mode.	None.
ERROR ON READ- ADDRESS:----	The disk parity error has been encountered while reading a record in disk mode.	None.
BEYOND END OF FILE- RECORD:----	A record beyond end of file has been requested to be read while in file mode.	None.
INVALID SECTOR- ADDRESS:----	A sector address beyond the last physical sector of a disk has been specified to be read while in disk mode.	None.

## DD (Disk Dump)

This utility allows the operator to back-up (“STORE”) one or more files from a disk to one or more disks (cartridge or BSMD) whose capacity is less than the originating media, and later “RESTORE” these files to their original state.

### Store Function:

Format:

```
DD STORE [disk-name /] file-name or group-name TO [disk-name /] file-name or group-name
```

The STORE function allows the operator to store a file or group of files between two disk media, where the size of the file may be greater than the physical size of the disk to which the file is being copied (destination disk). The files to be stored are specified by “file-name” or “group-name” and are taken from the system disk unless “disk-name” is specified. Additional files or groups of files may be specified, separated by a comma.

When the original destination disk is filled, the utility will request the name of the next disk to be used. The disk name is accepted from the operator who should then power off the drive using the “PO” intrinsic, insert the new disk into the drive, and use the “RY” intrinsic to allow the utility to continue. When all the files have been copied, the original destination disk must be re-inserted before DD will go to End of Job.

STORE re-computes block sizes for optimization and to allow a file to be copied to multiple disks. As a result, these files will not be usable as in their original form. The files must be RESTORED.

Examples:

To store a file called PR200 from the disk PRI to the disk PRBU:

```
DD STORE PRI/PR200 TO PRBU/PR200
```

To store a group of files beginning with the letters “PR” from the disk PR2 to the disk PRBU:

```
DD STORE PR2/PR= TO PRBU/PR=
```

To store a file called PRCLEF from the system disk to the disk PRBU:

```
DD STORE PRCLEF TO PRBU/PRCLEF
```

The information needed to restore these files is kept in a file called “DDSTORE” which is created when DD begins processing. Because of this, DD may not be run twice to the same destination disk or the information required to restore the first set of files will be lost.

### Restore Function

Format:

```
DD RESTORE [disk-name /] file-name or group-name TO [disk-name /] file-name or group-name
```

This option allows the operator to restore a file or group of files between 2 disks where the disk from which the files are being copied (source disk) was generated by “file-name” or “group-name”. Files are copied to the system disk unless “disk-name” is specified. Additional files or groups of files may be specified, separated by a comma.

When all files have been restored, the utility will inform the operator what disk must be inserted next to continue the transfer. Power off the disk drive using the "PO" utility, insert a new disk, and use the "RY" intrinsic to continue the utility.

**Examples:**

To restore a file called PR200 from the disk PRBU to the disk PRI:

```
DD RESTORE PRBU/PR200 TO PR1/PR200
```

To restore a group of files beginning with the letters "PR" from the disk PRBU to the disk PR2:

```
DD RESTORE PRBU/PR= TO PR2/PR=
```

**Output messages:**

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
group-name - NO FAMILY MEMBERS FOUND	Specified group is not on disk.	Check input and re-enter if necessary; Check for correct disk.
FILE NAME TOO LONG	Specified file name is longer than 12 characters.	Re-input correctly.
file-name NOT STORED - ILLEGAL REQUEST	Attempt was made to store a system file (that is, MCP, interpreters, etc).	None.
ENTER NEW PACK ID FOR CONTINUATION OF TRANSFER	Program requests next disk for STORE continuation ACPT display.	Enter AX mix number of program and new disk name to which STORE will continue to copy.
INSERT PACK disk-name FOR CONTINUATION OF TRANSFER INSERT NEWPACK disk-name FOR CONTINUATION OF RESTORE	Continuation disks are required.	Respond to ACPT message with AX mix number disk-name. Power off (PO) disk currently filled, insert new disk, type "RY" to continue processing.
NOT FIRST PACK OF A STORED SERIES PACK disk-name REQUIRED	Source disk specified in RESTORE is not first disk of a stored series of disks.	Power off (PO) inappropriate disk, insert proper disk.
file-name STORED TO file-name file-name RESTORED TO file-name	STORE or RESTORE successful	None.

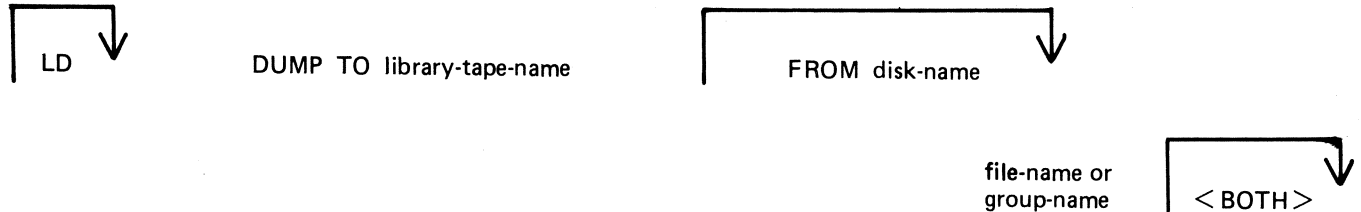
NOTE: Refer to "Common Utility Output Messages" for additional aid.



## DUMP (Dump Files to Library Tape from Disk)

This function, a part of the LD allows the operator to copy files from disk to library-tape.

Format:



If the <BOTH> option is used following a request to dump a keyfile, the associated data file will also be dumped following the keyfile on tape, provided the data file is on disk.

Examples:

To dump a group of files beginning with the letters "AR" from the system disk to ARTAPE:

DUMP TO ARTAPE AR=

To dump a file called ARTASK and a group of files beginning with the letters "DCS" from a disk called ARDISK1 to a tape called ARTAPE:

LD DUMP TO ARTAPE FROM ARDISK1 ARTASK DCS=

To dump a keyfile called AR200K and its data file from a system disk to ARTAPE:

DUMP TO ARTAPE AR200K <BOTH>

Since "DUMP" is a part of the utility "LD", "LD" is actually what will appear in a mix message. To discontinue the DUMP function, 'DS' mix number/LD must be used.

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
NO FILES IN THE FAMILY group-name ON DISK disk-name FOR DUMP	Specified group name not found on disk.	Check input and re-enter if necessary; Check for correct disk.
NO FILE file-name ON DISK disk-name FOR DUMP	Specified file was not found on disk.	Check input and re-enter if necessary; Check for correct disk.
file-name NOT DUMPED - IN OUTPUT USE- DUMP ABANDONED TAPE BEING PURGED.	File to be copied is in use and cannot be copied. If "DUMP ABANDONED" message is given, tape is purged and utility goes to EOJ.	None: utility will not dump this file but will continue with next file to be dumped.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
file-name NOT DUMPED - HAS BEEN REMOVED. DUMP ABANDONED - TAPE BEING PURGED.	File to be copied was removed between start of DUMP and time when file was to be copied to tape. Tape is purged and utility goes to End of Job.	None
file-name NOT DUMPED - HAS BEEN ALTERED - DUMP ABANDONED - TAPE BEING PURGED.	Contents of file to be copied were altered between the start of the dump and the time the file was to be copied to tape. Tape is purged and utility goes to End of Job.	None
file-name LOAD/ DUMP DISCREPANCY	End of File reached before expected. Disk File Header may contain errors.	None.
NO FILES TO DUMP	No files were found on this disk to copy to tape.	Check input and re-enter if necessary; Check for correct tape.
file-name DUMPED	DUMP successful	None.
DUPLICATE file-name ALREADY BEING DUMPED	More than one request was made to DUMP this file.	None.
file-name NOT DUMPED - DATA FILE NOT ON LINE	<BOTH> option was specified, but data file was not found on disk.	If specified data file dump is required, supply DUMP with backup copy of file if it exists.

Note: Refer to "Common Utility Output Messages for additional messages.

## ECMA.LD (LOAD/DUMP OF ECMA TAPE FILES)

This utility allows the operator to structure tape files according to ECMA BASIC and ECMA COMPACT systems as specified in the STANDARD ECMA-41 publication.

The ECMA tapes are treated as unlabelled tapes in the CMS system. The utility is initiated in two different ways for BASIC system and COMPACT system.

### BASIC INITIATION

Format (Disk to Tape copy)

```
ECMA.LD  DUMP  
         or  
         UNLOAD  FROM disk-name  file name
```

The files are copied from the disk to a purged tape. If the option UNLOAD is specified then the files copied are removed from the disk.

Format (Tape to Disk copy)

```
ECMA.LD LOAD  TO disk-name  RECORD number  BLOCK number
```

The files are copied from an ECMA BASIC tape to the disk specified by disk-name. The option RECORD followed by a number specifies the record size in bytes on the tape and BLOCK followed by a number is the blocking factor, that is, number of records per block.

If the RECORD and BLOCK options are used then the first file on tape is read according to the first attributes, the second file is read according to the second attribute etc. If the block size (that is, record size x number of records per block) exceeds 256 then an error message is issued and loading is not performed.

If RECORD and BLOCK options are not used then tape files are loaded as 256 byte records blocked 1.

The names of the ECMA tape files loaded to the disk become ECMA001, ECMA002,....ECMA00n.

#### Examples:

To copy files IN001, IN002 from the systems disk to a tape in ECMA BASIC format:

```
ECMA.LD DUMP IN001, IN002
```

To copy file AR030 from the disk ARDISK to a tape and remove after copy:

```
ECMA.LD UNLOAD FROM ARDISK AR030
```

To copy files PR200, record size 80 block 160, PR210, record size 60 block 180, to the system disk from ECMA BASIC tape on device CTA:

```
ECMA.LD LOAD RECORD 80 BLOCK 2, RECORD 60 BLOCK 3
```

The utility will display the following message:

```
mix no/ECMA.LD <14> WAITING UNLAB ECMATAP/NONE AT DEVICE REQUIRED
```

Then enter:

```
AD mix number CTA
```

The files PR200 and PR210 will be called ECMA001, ECMA002 on the disk.

## COMPACT INITIATION

Format (Disk to Tape copy)

```
ECMA.LD      DUMP
             or
             UNLOAD  TO tape name  FROM diskname  file name
```

The files are copied from the disk to tape. If the option UNLOAD is used then files copied are removed from the disk.

Format (Tape to Disk copy)

```
ECMA.LD      LOAD
             or
             ADD     FROM tape-name  TO disk name
```

The files are copied from an ECMA COMPACT tape to the disk specified by disk name. If any file is already present on the disk, it will be removed before the same named file is copied. The option RECORD followed by a number specifies the record size in bytes on the tape and BLOCK followed by a number is the number of records in a block.

If RECORD and BLOCK options are used then the first file on the tape is read according to the first attribute and the second file is read according to the second attribute. If block size (that is, record size x number of records per block) exceeds 256 then an error message is given and utility will terminate.

If RECORD and BLOCK options are not used then tape files are loaded as 256 byte records blocked 1.

The ADD option will copy a file to the disk only if that file is not already present.

Examples:

To copy files IN001, IN002 from the disk INDISK to a tape FRED in ECMA COMPACT format:

```
ECMA.LD DUMP TO FRED FROM INDISK IN001, IN002
```

To copy files IN010 from the system disk to a tape FRED and remove from the disk after copy:

```
ECMA.LD UNLOAD TO FRED IN010
```

To copy files from tape FRED on CTA which is created in ECMA COMPACT format files IN001, IN002 to the system disk:

```
ECMA.LD LOAD FROM FRED IN001 RECORD 80 BLOCK 2, IN002
```

The utility will display the following message:

```
Mix number/ECMA.LD <14> WAITING UNLAB ECMATAP/NONE AT DEVICE REQUIRED
```

The response should be:

```
AD mix number CTA
```

Output Messages:

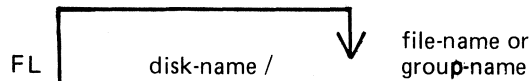
MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
NOT A BASIC ECMA TAPE	Loading is requested from a BASIC tape but the tape is not a BASIC format	Check the tape and re-input
NOT A COMPACT ECMA TAPE	Loading is requested from a COMPACT tape but the tape is not a COMPACT format	Check the tape and re-input
NO FILES IN THE FAMILY group-name ON TAPE tape-name FOR DUMP/LOAD/ADD/UNLOAD	Specified files are not present on the tape	Check the group name and/or the tape, and re-input
NO FILES IN THE FAMILY group-name ON DISK FOR DUMP/LOAD/ADD/UNLOAD	The specified group of files is not present on disk	Check the group name and re-input
NO FILE file-name ON TAPE tape-name/DISK disk-name FOR LOAD/DUMP/ADD/UNLOAD	The specified file is not present on tape or on disk	Check the file name and re-input
file-name REMOVED	This message is displayed for each file removed by a LOAD or UNLOAD	none
file-name LOADED	This message is displayed for each file added or loaded	none
file-name DUMPED	This message is displayed for each file dumped or unloaded	none
file-name NOT LOADED-ALREADY ON DISK	Self-explanatory	
file-name NOT FOUND	The file is not found on disk	Check input and re-enter
file-name HAS BEEN REMOVED - DUMP ABANDONED - TAPE BEING PURGED	The specified disk file to be copied to tape has been removed before the utility was able to copy it	none

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
file-name HAS BEEN ALTERED - DUMP ABANDONED - TAPE BEING PURGED	The contents of the file to be copied to tape were altered after the start of the dump and before the utility was able to copy it	none
file-name IN OUTPUT USE - DUMP ABANDONED - TAPE BEING PURGED	The file cannot be dumped to tape as it is being used as an output file	none
BLOCK COUNT ERROR	This is displayed when the BLOCK COUNT entry in the end-of-file label, end-of-track label, or end-of-volume label does not match with the actual number of blocks read during a LOAD or ADD	possible bad tape: retry with a different drive or with a backup tape
INCORRECT TAPE NAME	For a COMPACT format tape, the names in the initiating message and the tape label do not agree	Check the tape label and re-input
NOT FIRST PART OF FILE	A section number other than 1 was found when reading the first label on tape	
DATA BLOCK TOO LARGE	The block size on tape for a LOAD or DUMP is greater than 256 bytes	none
file-name DATA FILE NOT FOUND ON TAPE FOR LOAD	File specified for COMPACT format LOAD cannot be found	Check file name and re-input if necessary

# FL (Display File Attributes on Self-Scan)

This utility allows the operator to display detailed information about particular files or groups of files on disk upon the self-scan screen. The information given is similar to the LR utility.

Format:



The utility uses the following PKs when more than one file is specified:

PK1	PK2	PK3	PK4	PK5	PK6
page to next screen	—	—	—	—	EOJ

Examples:

To display information about all entries on the system disk:

FL =

To display information about a file called PR200 found on disk called PR2:

FL PR2/PR200

To display information about a group of files beginning with the letters "PR" found on the system disk:

FL PR=

Output format:

The utility will fill eight lines on the self-scan screen with information about one file.

line 1: FILE NAME (DDDDDDDD/FFFFFFFFFFFFF)

line 2: FILE TYPE (TTTTTTTT)

line 3: SIZE:ACTUAL (XXXXXXX) MAX (XXXXXXX)

lines 4,5: headings

line 6: DATE CREATED (YYDDD),LAST ACCESS DATE (YYDDD),  
RECORD SIZE (XXXXX), RECS/BLOCK (XXXXX)

line 7: AREA MAP (\*\*\*\*\*)

line 8: OVERFLOW ON DISK (DDDDDDDD)

Note that the OVERFLOW ON DISK will not be displayed if the file has no overflow areas allocated.

The first line contains disk name specified by DDDDDDD on which the file specified by FFFFFFFFFFFFFF resides.

In the second line the FILE TYPE entry will contain one of the following:

- SYSTEM (system file)
- CODE COMPILED YYMMDD (object code file and compilation date)
- DATA (normal data file)
- SRCELANG (source language file)

The third line displays the actual file size and maximum file size specified for the file.

The fourth, fifth, and sixth lines display the date of creation and the date the file was last accessed in YYDDD (Julian) date format. The record size displays the number of characters per record and RECS/BLOCK displays the number of records per block.

In the seventh line sixteen characters are displayed to show the allocation of the sixteen areas into which a file may be broken. Each character may be one of the following:

- \* unallocated
- B allocated on this disk
- O allocated on the overflow disk

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
FILE NAME = disk-name/file-name NOT FOUND IN DISK DIRECTORY	Specified file not found on this disk.	Check input and re-input if necessary; Check for correct disk.
disk-name/file-name NO FILES FOUND IN DIRECTORY FOR FAMILY	Specified group was not found on this disk.	Check input and re-input if necessary; Check for correct disk.

Note: Refer to "Common Utility Output Messages" for additional aid.



(3)

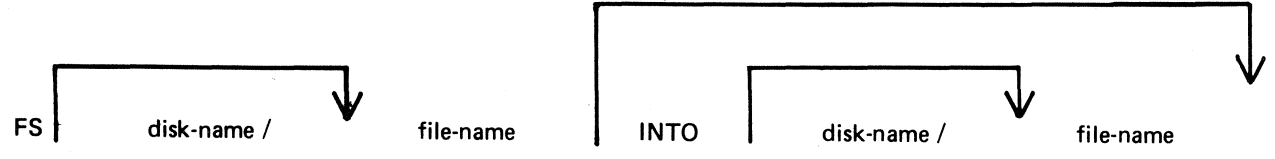
*Nei record cancellati, ci sono tutte FFF. ma continue ad esistere  
con FS -> si comprime eliminando finché rimane il record*

### FS (File Squash)

*Compattezza*

This utility allows the operator to remove all deleted records from a data file. Records are normally "deleted" (that is, hexadecimal @FF@ are written over the records) through an appropriate application program. The FS utility will remove these previously deleted records, allowing additional records to be added to the file.

Format:



The "file-name" identifies either a data file or a keyfile. If a keyfile has been specified, the name of the data file is obtained from the information held in the keyfile.

If a keyfile is specified, then the utility will reconstruct this keyfile so that it relates to the modified data file.

While the utility is processing, no other program may access the data file (or the keyfile if one is specified).

If only one file-name is specified and no other options are used, the file squash will be carried out in place, and no new data file will be created. If a keyfile was specified, then a new keyfile with the same "file-name" will be recreated by the SORT.

If 2 file-names are specified, the data file will be squashed into a new file, and the keyfile (if specified) will be recreated by the SORT. If the first file-name specifies a keyfile, then the new keyfile will have the name indicated by second file-name, and the new data file will have the name of the new keyfile name, with the letters "QQ" attached to the end of the name.

Examples:

To squash the file, PR200:  
FS PR200

To squash the file, PR200 and create a new file, PR200B:  
FS PR200 INTO PR200B

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
file-name SQUASHED FROM n RECORDS TO m RECORDS	FS successful. Original and resulting filesizes of the data file indicated. "n" and "m" are decimal numbers up to 7 digits each.	None.
KEYFILE file-name RECONSTRUCTED	"file-name" indicated by operator input identified by keyfile and FS has successfully squashed the data file and reconstructed the keyfile.	None.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
KEYFILE SORT FAILURE	SORT utility was not able to properly construct a keyfile.	None.
RECORD SIZE TOO GREAT	An attempt has been made to squash a file with a record size greater than 10000 bytes.	None.
ZIP TO SORT FAILURE - END OF JOB	When attempting to reconstruct the keyfile, the utility failed to execute SORT.	Ensure that SORT and SORTINTRINS are present on the disk, and re-input.

Note: Refer to "Command Utility Output Messages" for additional aid.

## ICMD (Industry Compatible Mini Disk Access):

This utility allows the operator to access industry-compatible mini disks (ICMD).

### Format:

```

ICMD LR disk-name
or
ICMD PG disk-name TO disk-name owner name
ICMD COPY disk-name/file-name ICMD TO disk-name/file-name FILESIZE number
or
ICMD COPY disk-name/file-name TO disk-name/file-name B ICMD
  
```

The first function of the utility (LR) allows the operator to print the disk directory of the ICMD. The utility will print a line of information for each file on disk.

The second form of the utility (PG) allows the operator to purge (erase) all files from an ICMD. The utility will replace all files by a single zero-length file called "DATA" to which is assigned all the disk space available to the user. By specifying 'TO', the operator may change the disk's name to the second name, and in this case the owner's name may be set or made blank.

The third form of the utility allows the operator to copy an ICMD file to a CMS file. The CMS file will have the largest block not exceeding 180. If the FILESIZE of the CMS file was not specified, it will be calculated from the header of the ICMD file. Note that when copying a Multi-Volume File (file which resides on more than one disk) the FILESIZE option will almost certainly be required.

The fourth form allows the operator to copy a CMS file to an ICMD file. The maximum record size of the CMS file must be 128 bytes (the ICMD sector size). The option 'B' allows the copying of any file without the loss of attributes, but it may be copied back by this utility only.

### Examples:

To print the disk directory of an ICMD disk called PR2:

```
ICMD LR PR2
```

To purge all the files from the ICMD called PR2 and name it as PR3:

```
ICMD PG PR2 TO PR3
```

To copy a file called PR200 from the CMS disk called PRI to an ICMD disk called PRBU:

```
ICMD COPY PRI/PR200 TO PRBU/PR200 ICMD
```

To copy a file called PRFILE from the ICMD called PRBU to the CMS disk called PRI:

```
ICMD COPY PRBU/PRFILE ICMD TO PRI/PR200
```

### Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
ENTER VOL-ID FOR "file-name" n	Program is about to display an ACPT requesting the next disk-name containing the Multi-Volume File "file-name". "n" is the sequence number to be used, or blank if none.	Enter appropriate disk-name.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
VOLUME n OF "file-name" OUT OF SEQUENCE	Sequence number "n" of "file-name" was not expected at this point. (Ex: program was expecting disk with sequence number of 2 and operator supplied disk with sequence number 4).	Remove inappropriate disk from disk drive (PO cannot be used with ICMD). Replace with disk having correct sequential number for this file.
DUPLICATE FILE "file-name"	File cannot be copied to ICMD as a file of the same name already exists. Disk cannot contain 2 files with the same name.	Copy this file while changing its name. For example, if file-name MYFILE is already on the ICMD, enter ICMD COPY DISK/MYFILE TO ICDISK/YOURFILE ICMD.
NO FREE LABEL ON "disk-name"	File cannot be copied to ICMD as its disk directory is full.	Replace ICMD with another ICMD having directory space. Re-attempt ICMD utility;
NO SPACE FOUND ON "disk-name"	File cannot be copied to ICMD as no unused space could be found.	FG ICMD and re-attempt copy; or select another disk on which to copy.

Note: If any of the above messages (including "disk-name" NOT FOUND and "file-name" NOT FOUND) is output in response to the initiating message, the utility will go to End of Job. If in response to an ACPT, the utility will repeat the message: ENTER VOL-ID FOR "file-name" "n".

BAD INDEX TRACK ON disk-name SECTOR number	ICMD has an error on Track 0 and cannot be used by this utility. Utility goes to EOJ.	Select another ICMD for this function.
RECORD SIZE OF file-name EXCEEDS 128	The file cannot be copied to the ICMD as its record size is more than 128, the maximum for an ICMD. Utility goes to EOJ.	Use MCP COPY utility to change record/block sizes. (EX: COPY CMSFILE TO CMSFILE RECORD 128, BLOCK 128) Then use ICMD COPY to complete transfer.

WV

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
NEXT SEQUENCE NUMBER OF file-name EXCEEDS 99	No more disks may be used, as the maximum for an ICMD is 99. Utility goes to EOJ.	Use MCP COPY utility to break file (see COPY utility). Then use ICMD COPY to complete transfer.
INPUT ERROR ON file-name SECTOR number	Irrecoverable error was found on reading sector "number" from the ICMD. Utility goes to EOJ.	File cannot be copied from the ICMD; use backup ICMD if available.
OUTPUT ERROR ON file-name SECTOR number	Irrecoverable error was found on writing sector "number" to the ICMD.	Disk cannot be used. Use another ICMD.
file-name TO file-name COPIED	ICMD COPY function successful.	None.
disk-name PURGED	ICMD PG function successful.	None.
file-name IS NOT TYPE SOURCE OR DATA	The requested file has the wrong file type for this use.	None.
file-name REMOVED	The CMS file was removed by the utility to make way for another file being copied from an ICMD.	None.

Note: Refer to Common Utility Output Messages for additional aid.

# IR (Initiate Log Recall)

(a function of SYS-SUPERUTL)

This function will initiate recall and go back in SYS-LOG files after skipping the number of entries specified by the operator (that is, 5 digit "offset") and display the required message.

Format:

IR offset

Examples:

To initiate recall after 12 entries and display the message on the console:

IR 12

To initiate recall of the message just given:

IR 1

Output messages

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
CANNOT LOCATE DESIRED LOG ENTRY	The log files have been transferred using TL utility	None.
	Offset is greater than number of entries in log file	Decrease the value of offset and re-enter.

## KA (Analyze Disk Space Assignment)

This utility provides the operator with a map of all space used on disk by specific files, or available for other use. The printout is in ascending disk address order in terms of areas and their assignment.

KA is capable of analyzing space assigned to one or more files, one or more groups of files, or the available areas.

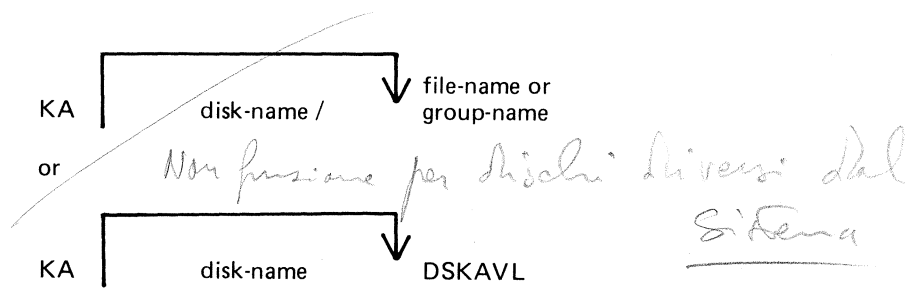
Special reporting is given if the group identifies all files on the disk (that is, disk-name/=). In addition to an analysis of the areas allocated to each file, this report will show the space assigned to the disk directory, temporary, available, bad, and missing areas. The temporary areas are those which are allocated either to temporary files or to the virtual memory.

If files are created, extended, or deleted by the system during the processing of KA the map will not be accurate. It is therefore necessary that KA be run only when no other programs are in the mix.

The analyzed output will be to a line or console printer, and will print the areas in ascending disk address order associating with each area its first sector address, its length in sectors, and its status. The status will be either allocated, available, temporary, bad, or missing. If the area is allocated, the file name of the file to which the area is assigned will also be listed. If a particular file or family is not on-line, then this is indicated on the printout.

If the option DSKAVL is selected, then an analysis of the available areas on the disk specified by "disk-name" (or system disk if no "disk-name" was specified) will be printed.

### Format:



### Examples:

To analyze disk space assignments of all files on system disk:

KA =

To analyze disk space assignments of all files on the disk called PR2:

KA PR2/=

To analyze disk space assignments for a group of files beginning with the letters, "PR" on the system disk, and a file called PR200 on a disk called PR2:

KA PR= PR2/PR200

To analyze available areas on the disk called PRBU:

KA PRBU DSKAVL

**Output format:**

Six columns of information are output. The column headings, the format of the values these columns contain, and the significance of these values are as follows:

HEADING -----	VALUE ----	SIGNIFICANCE -----
AREA ADDRESS	8 digits 6 hex. digits	Sector address of start of area
AREA LENGTH	8 digits 6 hex. digits	Number of sectors in this area
STATUS	9 characters	See Note 1
FILE NAME	12 characters	See Note 2

Note 1: The status will be one of AVAILABLE, TEMPORARY, BAD, or \*MISSING\*, depending on whether the area is available, allocated to a file, denoted as temporary, unusable, or lost.

Note 2: If the area is ASSIGNED, then this field will contain the identifier of the file residing in the area. If a file belongs to a pseudo disk its disk name is also listed. Otherwise it will be blank.

The status \*MISSING\* occurs if an area is not referenced from anywhere within the file directory or available table. This may be because the area is in fact lost, or because existing files have been opened, have had further areas allocated to them and are still open during the processing of KA.

If fixed disk is being used, three areas are reserved for MTR purposes with the status marked as "BAD". The area lengths are 256, 128, and 128 sectors respectively.

**Output messages:**

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
*AREA APPARENTLY ASSIGNED TWICE*	Area is contained partly or completely in an area previously listed.	Notify field engineering. DO NOT USE DISK. It may be possible to recover any file not involved, using COPY.
*AREA ASSIGNED BEYOND MAXIMUM ADDRESS*	Area is assigned beyond the addressable space.	Notify field engineering. DO NOT USE DISK.
NOTE: OUTPUT FILES ON DISK WERE OPEN DURING THIS EXECUTION OF KA	Files were open on disk while KA was processing. Other programs may have been in the mix when KA was processing.	KA printout may not be completely accurate. Begin KA when no other programs are in the mix.
input NOT FOUND IN DIRECTORY ON THIS DISK	Specified file is not on disk.	Check input and re-input if necessary; Check for correct disk.



8

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
input NO FILES IN DIRECTORY FOR THIS FAMILY	Group of files specified is not on disk.	Check input and re- input if necessary; Check for correct disk.
TABLE SIZE EXCEEDED	Number of lines of output required is greater than KA permits. Maximum permitted is about 141 pages of output. KA ends.	None.
NO OUTPUT GENERATED BY KA	Disk directory cont- ains no file names to print.	Check for correct disk; Check input and re-input if necessary.

Note: Refer to "Common Utility Output Messages" for additional aid.

## KEY.CHECK

This utility allows the operator to check and print the information on the validity of keys in an indexed pair of files.

Format:

KEY.CHECK disk-name/ keyfile-name <BOTH>

This utility does not provide the \* <file name> option in the initiating message.

During the execution of the utility the operator is informed if the data file has been updated via another key file or parity errors have occurred on the data file since creation of the specified key file. The checking is performed in two ways, Key file to Data file check, and Data file to Key file check.

In Keyfile to Data file checking, the key value field of each entry in the key file are compared with the key field in the corresponding record of the data file. This comparison will detect any changes to the keys in the data file records and disused entries in the key file.

In Data file to Key file checking, the key field of every non-deleted record in the data file will be checked to have an entry in the key file. The record written to the data file via another key file and records with invalid keys will be detected, as they will have no entry in the key file.

If the <BOTH> option is specified in the initiating message, then Key file to Data file and Data file to Key file will be checked otherwise only Key file to Data file will be checked.

The utility will terminate if a parity error is encountered on the key file.

If the Generation number of the key file differs from that of the data file then a warning is printed. (Generation number is a field in the File Parameter Block, refer to MCP manual).

The Generation number of the key file will be modified to that of the associated data file on completion of a Key file to Data file check provided that the following conditions are satisfied:

1. Every key entered in the key file matches the key field in the associated data file record.
2. Every non-deleted data file record has a key entry in the key file.
3. The number of matched key entries in the key file is equal to the number of data file records with a key entry in the key file.
4. There are no parity errors on either the key file or the data file.

Output Format:

The output format is self-explanatory.

Examples:

KEY.CHECK PQR

Check the key file PQR, performing Key file to Data file check only.

KEY.CHECK PQR <BOTH>

Check the key file PQR, performing Key file to Data file check as well as Data file to Key file check.

Output Messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
file-name IN USE	Another program is using the keyfile	Wait until job using keyfile has closed file, then re-input
file-name IS NOT A KEYFILE	Self-explanatory	Check the file-name and re-input
file-name IS A NULL-KEY KEYFILE	The specified file is a tag-file	See SOG section on the SORT for meanings: this utility cannot check tagfiles
file-name <DATAFILE> NOT FOUND	The data file corresponding to the keyfile cannot be found	(1)Enter LR disk-name/keyfile-name. (2)Check the disk name of the data file in LR listing. (3)If the disk file name is the same as the keyfile then copy data file from backup and re-input. (4)If the dis name is different then load that disk and re-input.
file-name <DATAFILE> IN USE	The corresponding data file is being used by a program	Wait until the other job has closed the data file, then re-input
file-name INCOMPATIBLE WITH KEYFILE	The key-offset plus key-length in the KFPB of the specified keyfile is greater than the record size of the data file	none
READ ERROR ON KFPB OF file-name	Parity errors have been encountered: the utility cannot continue.	(1)Attempt to copy files to another disk (2)Run CHECK.DISK to check the integrity of the disk
READ ERROR ON KEYFILE file-name	same as above	same as above

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
UNABLE TO OPEN file-name <INDEXFILE>	<BOTH> has been given in the initiating message and the utility is unable to open the keyfile and datafile as an indexed pair after completion of the keyfile-to- datafile check	none

Note: Refer to "Common Utility Output Messages" for additional aid.

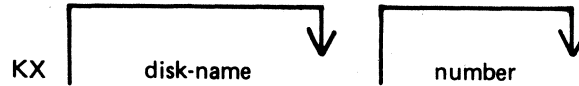
# KX (Disk Allocation Information)

*In whole file per file*

(a function of SYS-SUPERUTL)

This function will allow the operator to display the name of the first file found on the disk specified by "disk-name" (or on the system disk if no "disk-name" is specified) whose total number of sectors allocated is equal to or greater than "number" (assumed zero if not specified).

Format:



After each display, which will include the information of the current numbers of temporary and available sectors, the KX function of SYS-SUPERUTL remains available, waiting for one of the following input responses:

- A call on any other function of SYS-SUPERUTL: this will terminate KX.
- KX or KX NEXT To display the next file name, if any, otherwise KX will go to END.
- KX RM or KX REMOVE To remove the file whose name has just been displayed.
- KX END To terminate KX.

Examples:

To display the name of the first file on the system disk whose size is equal to or greater than 250 sectors:

KX 250

To display the name of the next file whose size is equal to or greater than 250 sectors:

KX NEXT

To remove the file just displayed:

KX RM

To terminate KX:

KX END

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
END KX	KX successful	None.
filename - number SECTORS; AV. number, TEMP (or TEMPORARY) number	Normal KA display showing current number of sectors, available and temporary on disk.	None.
input IS AN UNACCEPT- ABLE RESPONSE FOR KX	Typing error.	Check input and re-enter.

See SYS-SUPERUTL messages also.

10

## LB (Look Back in Log)

(a function of SYS-SUPERUTL)

This function will Look Back to continue recall in the direction of earlier messages with a screenful of messages. If the serial printer (SPA) is used as the console, then the function will display a number of messages calculated by the length of messages and width of console.

Format:

LB

Example:

To look back and display the messages:

LB

LB can be initiated only after IR, LB, and LF.

Output messages

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
IR REQUIRED BEFORE LB CAN BE DONE	IR function was not initiated before LB.	Initiate IR then re-input LB.
CANNOT RECALL LB BEYOND THIS POINT	LB has reached the beginning of this log file	None

## LD (Tape Library Utility)

This utility allows the operator to maintain library tapes. It is divided into the following four separate "sub-programs" (functions):

- ADD (tape-to-disk file copy)
- LOAD (tape-to-disk file copy; duplicates are removed from disk)
- DUMP (disk-to-tape file copy)
- UNLOAD (disk-to-tape file copy; files are deleted from disk after being copied to tape)

On the B 80 these four functions can be invoked directly: the MCP will recognize that they are part of the LD program and load LD from the system disk, passing LD the appropriate information. For example, the input

DUMP TO ARTAPE AR=  
LD DUMP TO ARTAPE AR=

If LD does not reside on the system disk, the user-disk name and 'LD' must be specified.

Should the operator request a "mix message" (see MIX intrinsic) while any of the 4 functions are running, "LD" (not the name of the specific function) will appear in the mix.

Similarly, to discontinue any of the four functions, a message of: "DS mix number/LD" will be required.

Detailed descriptions of ADD LOAD, DUMP, and UNLOAD and associated output messages are provided under the name of the function.

11

## LF (Look Forward in Log)

(a function of SYS-SUPERUTL)

This function will look forward to continue recall in the direction of later messages with a screenful of messages. If the serial printer (SPA) is used as the console, then the function will display a number of messages calculated by the length of the messages and the width of the console.

Format:

LF

Example:

To look forward from last recall and display messages:

LF

LF can be initiated only after IR, LB, and LF.

Output messages

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
IR REQUIRED BEFORE LF CAN BE DONE	IR was not initiated before LF	Initiate IR and then re-enter LF
CANNOT RECALL LF BEYOND THIS POINT	LF has displayed the most recent entry in the log file	None

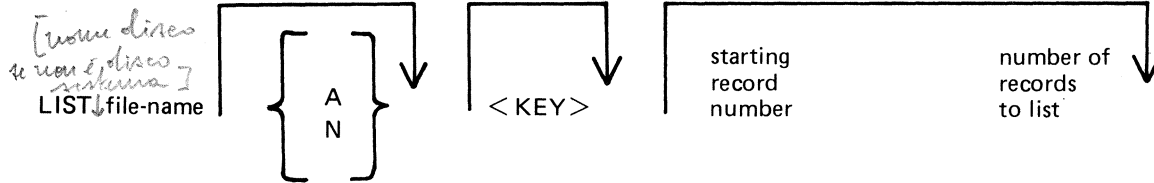


M

## LIST (File List)

This utility allows the operator to list in whole or in part files on any CMS device. Output will be either to the line printer or to the console printer.

Format:



If the "A" option is chosen the file will be listed in alpha characters. The "N" option will list the file entirely in hexadecimal. If neither the "A" nor "N" options are selected, the file will be listed in both alpha and hexadecimal.

If the file to be listed is a keyfile, the utility will list the associated data file in the order of the keyfile unless the <KEY> option is specified. When the <KEY> option is used, the utility will list the keyfile itself.

The operator may also list selected parts of a file by specifying the relative record number at which printing should begin and the number of records to be printed from that point.

Examples:

To list the records of a file called PROGSRC as alpha:

```
LIST PROGSRC A
```

To print the first record only of a file called PR200 in hexadecimal:

```
LIST PR200 N 1 1
```

To list records 100 through 149 of PROGSRC as alpha:

```
LIST PROGSRC A 100 50
```

To List Keyfiles:

Assume there is a keyfile called PR200K which refers to a data file called PR200.

The statement

```
LIST PR200K N <KEY>
```

 will list all records of the keyfile PR200K in hexadecimal.

The statement

```
LIST PR200K N
```

 will list all records of the data file, PR200 in keyfile order in hexadecimal.

## Additional Capabilities

Further features of this utility are summarized in the railroad chart given in Figure 4-3, which gives the complete input specifications.

Non-disk files

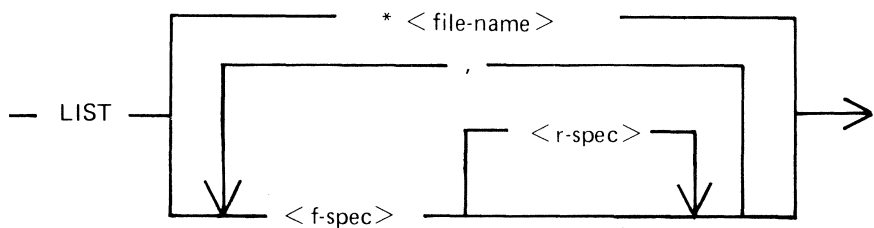
Files on media other than disk may be listed. Abbreviations for valid devices are as follows:

MTP - magnetic tape or cassette

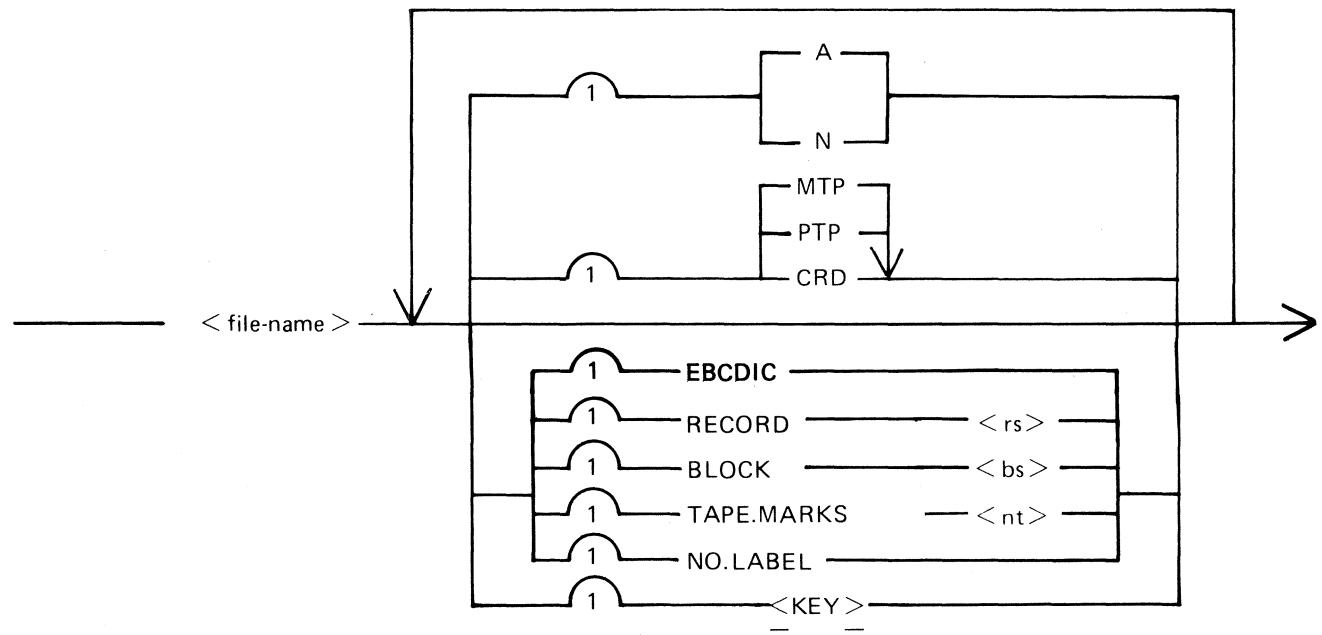
CRD - punched cards

PTP - paper tape

12



<f-spec> is defined as :



<r-spec> is defined as :

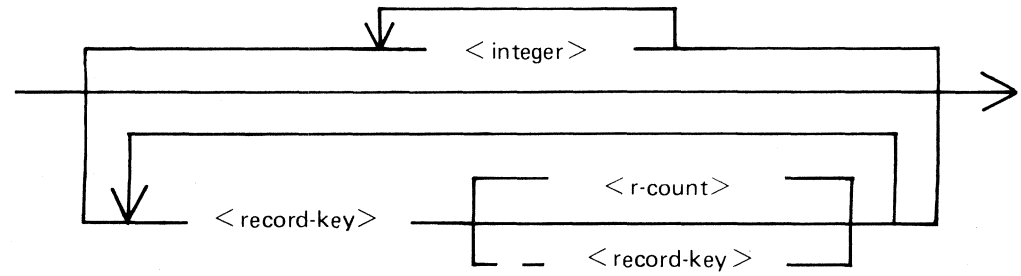


Figure 4-3. Railroad Chart for List Utility

**Examples:**

To list a cardfile called PRFILE in alpha:  
LIST PRFILE CRD A

To list the first 10 records of a CMS labelled magnetic tape called PRTAPE:  
LIST PRTAPE MTP 1 10

(Note: this assumes record size of 180 bytes). The tape or cassette to be listed should be a tape created by the COPY utility. Library tapes and non-CMS tapes should be treated as unlabelled (see below).

**Unlabelled tapes**

Input tapes having no CMS labels ("unlabelled" tapes) may be accessed by the LIST utility.

The NO.LABEL option allows the listing of unlabelled files. Upon recognizing an unlabelled file, the MCP will print a "DEVICE REQUIRED" message. The operator must then respond with an appropriate "AD" input message (see "AD") to identify the unlabelled file.

The end of file recognition for unlabelled files is determined by tapemark count. The TAPE.MARKS option allows the operator to specify the total number of tapemarks which will indicate end of file to the utility when listing an unlabelled file. The default value is 2. Each tape mark which is encountered will contribute to this total. Therefore, a standard labelled CMS file will be listed up to, but excluding, the trailing label if NO.LABEL and 2 tapemarks are specified. (A labelled CMS file consists of "Label, Tape mark, data, tape mark, label"). The operator must be aware of the format of any file which is to be listed when using the NO.LABEL option.

If the RECORD size is not 180 bytes, refer to the section on Record/Block modifications.

**Example:**

To list the first file of a magnetic tape with non-standard label (the format being: label, tapemark, data, tapemark):

LIST TP MTP NO.LABEL TAPE.MARKS 2

Note: MCP will issue a message asking for unlabelled tape TP. Operator must respond with "AD" input. Additionally, the first line of the listing contains a list of the non-standard label.

**Record and block sizes**

The listing is record-oriented. The following record sizes are assumed:

Disk = (Disk File Header) from file itself

Labelled tape/cassette = from tape label

Unlabelled tape = 180 bytes

Cards = 80 or 96 bytes depending on device.

If different values are required Record and Block sizes may be specified.

**Example:**

To list an unlabelled tape containing 10-byte records with 10 records per block:

LIST TP MTP NO.LABEL TAPEMARKS 2  
RECORD 100 BLOCK 1000

If EBCDIC is specified, the input will be translated from EBCDIC coding, otherwise ASCII is assumed.

For magnetic tape or cassette files the record size must be specified if it is greater than 1024 characters, otherwise the utility will not be able to read this file and therefore no list will be produced. If the record size is specified and no block size is specified then the block size will be set to the same as the record size. For unlabelled files the default record and block sizes are 180 each.

Note: Care should be taken to ensure that the record and block sizes specified are compatible with the physical block size on the tape. The block size specified must be an integer multiple of the record size. The utility will attempt to identify inconsistencies when using labelled CMS files. Any inconsistency not isolated by the LIST will cause MCP to discontinue (DS/DP) the utility.

**Selected file list**

More than one selected portion of the input file may be listed. Pairs of numbers may be specified within each pair the first number specifies a relative record number and the second specifies number of records to be listed. If an extra number is specified the last number specifies listing from that record to the end of file.

**Example:**

To list records 100 to 149, 300 to 499, and 1000 to end of file.  
LIST FILE1 100 50 300 200 1000

**Selected indexed file list**

For indexed files, listing of records can be selected based on content of the key. There are 2 options: the number of records can be specified or an ending key value.

**Examples:**

PQR is a keyfile containing personnel records. To list 15 records from the corresponding data file starting from the record with personnel number 01786:  
LIST PQR 01786 15

Using same keyfile to list all data records from personnel number 01786 to 18000:  
LIST PQR 01786 - 18000

Note: the second option is specified by the hyphen in the LIST statement. Note that at least one space is required before and after all key values (personnel numbers in this case).

**Output messages:**

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
number number IN filename NOT LISTED	Error found in pair of numbers for selected record list option. One record number in the pair indicates a section of the file at a lower file address than a previously specified section. This pair is ignored by LIST.	Check input and re-enter.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
filename EXHAUSTED DURING number number	End-of-file was encountered while the section of the file indicated was being copied.	Check input and re-enter if necessary.
filename EXHAUSTED DURING RANGE record-key record- key or filename EXHAUSTED DURING RANGE record key number	End-of-file encountered while section of file indicated is being listed; or no records were found in range of the 2 record keys.	Check input and re-enter if necessary.
NO RECORDS FOR LISTING FROM filename	Specified file contains no records to list.	Check for correct file-name.
filename NOT ACCEPT- ABLE - RECORD SIZE OF number EXCEEDS MAXIMUM FOR THIS RUN - RESUBMIT	LIST has encountered a file within a record size greater than expected. This can happen if a magnetic tape file with a record size greater than 1024 characters is submitted to the utility without the record size being properly specified in the initial input.	Check input and re-enter.
KEYFILE file-name OR DATA FILE NOT FOUND	Utility cannot locate keyfile or data file pertaining to the keyfile specifications.	Check for correct medium;
BAD ATTRIBUTE SPECIFIED	The number specified after BLOCK is not an integer multiple of the number specified after RECCRD.	Correct the input and re-input.
PERMANENT ERROR ON INPUT FILE file- name (NO. number or KEY key)	The irrecoverable error, e.g. parity error has been found.	None.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
SELECTIONS OF RECORDS FROM number IGNORED	The file reached end before the record selected.	None.
RECORD SIZE TOO GREAT	The specified record size is greater than 10000.	None.
INVALID CHAR. IN IDENTIFIER file-name - WARNING	A tape with an illegal name is specified.	Correct the name and re-input.
KEYFILE file-name OR DATA FILE IN USE	The specified keyfile and its data file are on line, but one or both are in use by a task which will not permit sharing.	Wait until the task using keyfile or data file goes to end of job.

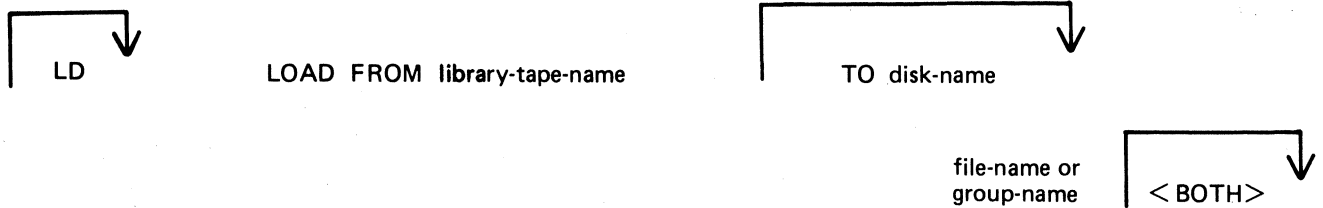
Note: Refer to "Common Utility Output Messages" for additional aid.

T A P E

## LOAD (Load Library Tape Files to Disk)

This function, a part of the LD utility, allows the operator to copy files from a library tape to disk. As files are copied to disk, any duplicate files will be automatically removed from disk by the function.

Format:



If the <BOTH> option is used immediately after a request to copy a keyfile, the associated data file will also be copied, provided that the data file does not precede the keyfile on the library tape. The keyfile will be altered to refer to the disk which now holds the data file (rather than the disk from which the data file was dumped).

Examples:

To copy all files from a tape called PRTAPE to a system disk:

```
LOAD FROM PRTAPE =
```

To copy the file called AR300 from a tape called ARTAPE to a disk called ARBU:

```
LOAD FROM ARTAPE TO PRBU AR300
```

To copy files called DCSTSK, and PRTASK from a tape called PRTAPE

```
LD LOAD FROM PRTAPE DCSTSK PRTASK
```

To copy from a tape called PRTAPE the keyfile called PR200K and its associated data file to the system disk:

```
LOAD FROM PRTAPE PR200K <BOTH>
```

Since "LOAD" is a part of the utility "LD", "LD" is actually what will appear in a mix message. To discontinue the LOAD function, "DS mix number/LD".

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
file-name LOADED	LOAD successful	None.
file-name REMOVED	LOAD successful: original file on disk was removed.	None.
library-tape-name NOT A RECOGNIZED DUMP TAPE	Tape does not have a recognizable header.	Provide utility with correct tape; DS the LD utility

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
NO FILES IN THE FAMILY group-name ON TAPE library-tape-name FOR LOAD	Specified group was not found on disk.	Check input and re-enter if necessary; Check for correct tape.
NO FILE file-name ON TAPE library-tape-name FOR LOAD	Specified file was not found on tape.	Check input and re-enter if necessary; Check for correct tape.
file-name LOAD/DUMP DISCREPANCY	End of File encountered before expected. Disk File Header may contain errors.	None.
NO FILES TO LOAD	No files were found on this tape to copy.	Check input and re-enter if necessary; Check for correct tape.
file-name DATA FILE NOT FOUND ON TAPE FOR LOAD	Data file for this keyfile does not follow on tape. It cannot be copied. Keyfile was copied.	Check if data file appears before key file on tape (use TAPEPD); if so, load full tape to disk.

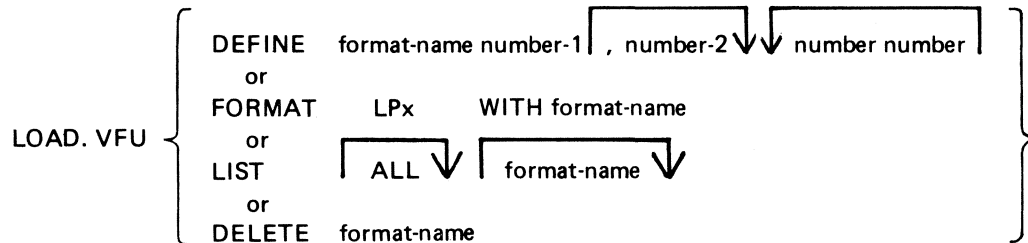
Note: Refer to "Common Utility Output Messages" for additional messages.



## LOAD.VFU (Load Vertical Format Unit)

This utility allows the operator to define the page format on a line printer that contains a soft vertical format unit.

Format:



The utility may be used to define vertical format unit formats and store them in a library file SYSVFU.LIB. These formats can be subsequently selected by name to be loaded to the specified printer (type A 9249-30/50). The utility zips the VF intrinsic which performs the actual loading of a format string.

When using the DEFINE function, if the file SYSVFU.LIB does not already exist, the utility will create this file and an entry will be made for the format being defined. Otherwise, the file will be updated to contain the newly defined format. The number-1 is the page height in lines and number-2 is the end of page line. The pair of numbers consists of a channel number (2-11) followed by a 'skip to line' number (not greater than the page height).

Example:

```
LOAD.VFU DEFINE PAYROLL 66, 60, 2 10, 4 20
```

This format is equivalent to:

```

Page Height = 66
End of Page = 60
Channel 2 has associated line number 10
Channel 4 has associated line number 20
  
```

This format is also equivalent to figure 4-4, which illustrates a paper vertical format tape on other line printers.

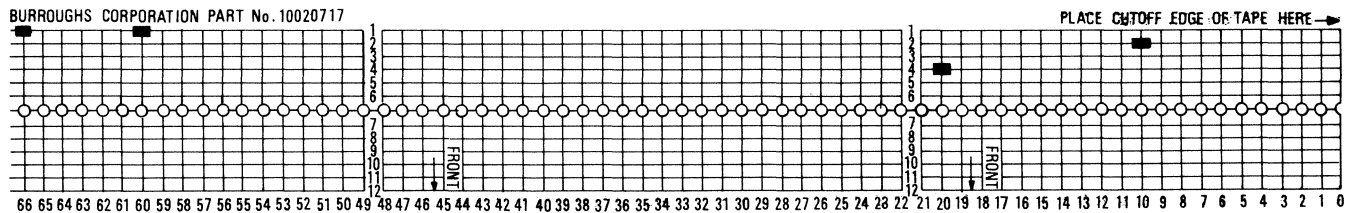


Figure 4-4. Paper Vertical Format Tape

Limitations:

The utility does not check the following:

1. For each channel, no more than 4 line numbers can be specified.
2. Page height defined should not be greater than 94 lines.

The FORMAT function allows the operator to load a predefined VFU format to a specified printer. The printer must be ON LINE and NOT IN USE.

Example:

To load the format PAYROLL defined earlier on LPA:

LOAD.VFU FORMAT LPA WITH PAYROLL

The LIST function has three possible options:

LOAD.VFU LIST

will list all format-names defined in SYSVFU.LIB;

LOAD.VFU LIST format-name

will list the format string of specified format-name as defined in SYSVFU.LIB;

LOAD.VFU LIST ALL

will list all format-ids and strings as defined in SYSVFU.LIB.

The DELETE function is used to delete a predefined VFU format from the SYSVFU.LIB file. If the format being deleted is the only defined format in SYSVFU.LIB, then the SYSVFU.LIB file will be removed.

Example:

To delete predefined VFU format PAYROLL from the SYSVFU.LIB file:

LOAD.VFU DELETE PAYROLL

NOTE: The default values of page height and end of page are 66 and 60 respectively. Non-default values require that the format should be loaded prior to the execution of the program.

Output Messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
SYSVFU.LIB NOT FOUND	Self-explanatory	Use DEFINE function to create library file
SYSVFU.LIB IN USE	Another copy of LOAD.VFU is running	Use only one copy of the utility
READ ERROR ON SYSVFU.LIB	Parity error on the library file	
WRITE ERROR ON SYSVFU.LIB	Parity error on the library file	
format-name NOT FOUND IN SYSVFU.LIB	Requested DELETE, FORMAT or LIST cannot find specified format-name	Check input and re-enter if necessary: use DEFINE function to create format-name
format-name ALREADY DEFINED IN SYSVFU.LIB	Cannot DEFINE two formats of the same name	Re-input using different format-name
format-name DEFINED IN SYSVFU.LIB	DEFINE function has been successful	none
format-name DELETED FROM SYSVFU.LIB	DELETE function has been successful	none

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
SYSVFU.LIB DELETED	A DELETE function has deleted the only remaining format in the library file, and the file has been removed	none
PRINTER peripheral FORMATTED WITH format-name	FORMAT function has been successful	none
PRINTER peripheral NOT FORMATTED WITH format-name	The utility has found the specified format in the library, but the zip of the VF intrinsic was unsuccessful (this will be preceded by output from VF)	refer to the VF error message

## LR (List Directory)

This utility allows the operator to print detailed information about particular files or groups of files on disk.

If a file has areas on an associated overflow disk, then the disk name of the overflow disk is printed beside each relevant area address and size. Note that the addresses for the areas on an overflow disk are not necessarily correct.

If a particular file or group is not found on a specified disk, this is indicated on the listing.

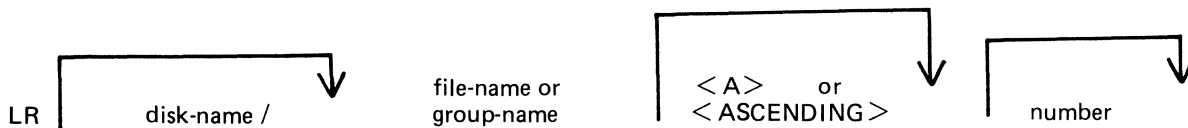
If "<ASCENDING>" or "<A>" is selected, the utility will print the information requested in ascending order of file-names.

If "number" is specified after the LR of an entire disk (that is, LR ARDISK2/=), then LR will only print information about those files whose total number of sectors allocated is greater than "number"; this will be followed by a listing, with totals, of all available and temporary areas on the disk.

An output line concerning a keyfile will normally be followed by a second line showing the name of the data file to which this keyfile points and the key offset and length.

The heading lines printed at the top of each page will provide a good deal of information about the disk itself.

### Format:



### Examples:

To print the entire directory of the system disk:

LR =

To print the entire directory of ARDISK2 in ascending order:

LR ARDISK2/= <A>

To print information about the file called "AR200" and a group of files beginning with the letter "C" only:

LR C=, AR200

To print information only about files on the system disk which have been allocated greater than 1000 sectors:

LR = 1000

### Output format:

Fourteen columns of information will be output to the printer for each disk for which information is requested. The column headings, the format of the values these columns contain, and the significance of these values are as follows:

HEADING -----	VALUE -----	SIGNIFICANCE -----
FILE NAME	12 Characters	File name
ACTUAL SIZE	7 digits	Number of records currently contained in file
MAXIMUM SIZE	7 digits	Maximum number of records which file may contain
RECORD SIZE	5 digits	Number of bytes per record
RECORDS BLOCK	5 digits	Number of records in each block
CREATED	5 digits	File creation date (Julian YYDDD)
ACCESSED	5 digits	Last access date (Julian YYDDD)
FILE TYPE	8 characters	See Note 1
NO. AREAS	2 digits	Number of areas currently allocated
AREA ADDRESSES	8 digits	See Note 2
AREA SIZES	6 hex. characters	See Note 2
OVERFLOW DISK	8 digits	See Note 2
	6 hex. characters	
	7 characters	Name of overflow disk Blank if not an overflow area

Note 1: The FILE TYPE will be one of the following:

- DATA - normal data file
- SRCELANG - source language file
- SRCELIBR - source library file
- CODE - object code file, followed by compilation date (YYDDD)
- KEY - key file
- SYSTEM - system file (for example, MCP,interpreters)

Note 2: For each file the area addresses and sizes of allocated areas will be printed in these columns. For areas on an overflow disk the overflow disk name will follow the size.

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
TABLE SIZE EXCEEDED	The number of file-names to be sorted by LR in an ASCENDING or A request has exceeded the maximum permitted (254).	None.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
input - NOT FOUND IN DIRECTORY OF THIS DISK	File specified is not on the disk	Check the correct disk and re-enter if necessary
input - NO FILES IN DIRECTORY FOR THIS FAMILY	Group of files specified is not on disk	Check input and re-enter if necessary. Check for correct disk.

Note: Refer to "Common Utility Output Messages" for additional aid.

## MODIFY (Program Code File Modification)

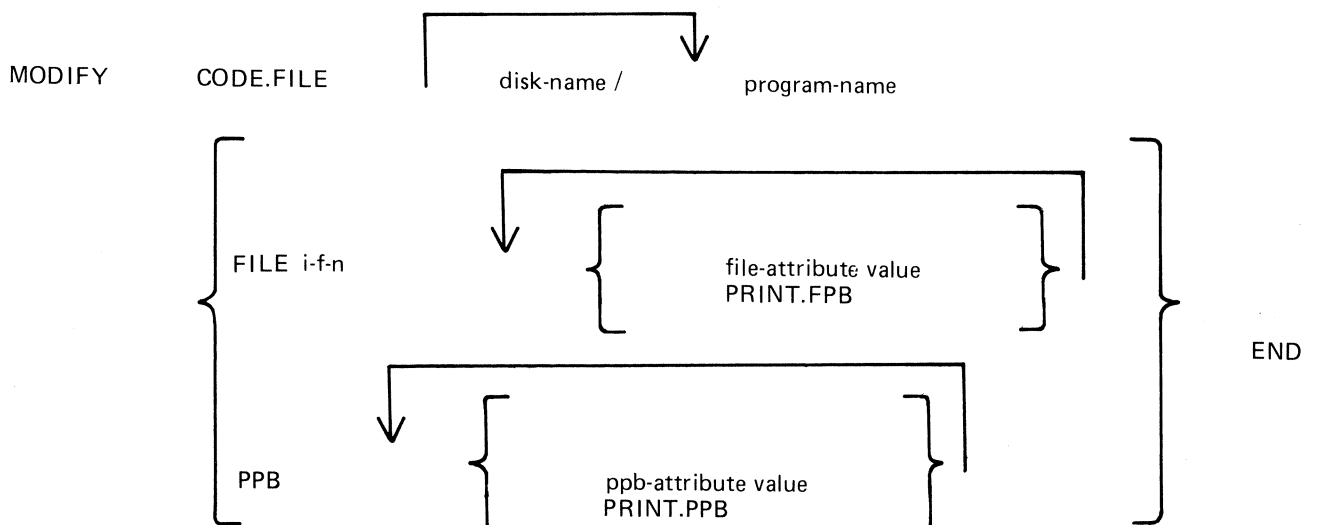
This utility allows the changing of a number of file attributes within the file parameter block (FPB) and program attributes within the program parameter block (PPB) of a code file. It should not be used unless the meaning of each attribute is thoroughly understood. Refer to the CMS MCP manual for more information on FPB and PPB formats.

The utility operates in an interactive manner using a console file if no further information is provided when initiating the utility, thus:

MODIFY

For details of the interactive mode, see later. Specifications can be entered when starting the utility. The name of the code file to be modified is preceded by the keyword "CODE.FILE". The word "CODE.FILE" can be omitted from the first element of the initiating message. Following the code file name is either the keyword "FILE" to enable file attributes to be modified, or the keyword "PPB" to enable program attributes to be changed. The file whose attributes are to be changed is specified by the internal file name (i-f-n) as given by the program source code listing. The i-f-n is determined by the programmer. Additional keywords are "PRINT.FPB" and "PRINT.PPB" to print the complete FPB and PPB respectively. The complete specifications to the utility are terminated by the keyword "END".

Format:



The commas are optional, but may be used to improve readability. See later for the list of attributes and allowable values.

Examples:

To modify the value of FID (file-id) and change the device kind of a file whose internal name is INFILE in a program code file COPY on disk SYS2:

```
MODIFY CODE.FILE SYS2/CCPY, FILE INFILE FID CARDS DEVICE CR, END
```

To change the value of CONTROL.STACK to 50 in code file AR768 on disk AR1, and print the resultant PPB:

```
MODIFY AR1/AR768 CONTROL.STACK 50 PRINT.PPB END
```

## Interactive Mode

If no initiating specifications are given, PKs 1 to 6 are lit for various functions.

PK1	PK2	PK3	PK4	PK5	PK6
help	modify PPB	modify FPB	specify code file	print FPB or PPB	ECJ

Pressing PK1 gives a display of the meanings of the 6 PKs, as shown above, followed by the request  
CODE.FILE?

Enter the code file name, followed by OCK1. The utility requests  
SELECT FUNCTION

and lights appropriate PKs. While any relevant PK is lit, the corresponding function can be started.

If PK2 (modify PPB) or PK3 (FPB) is depressed, the utility requests  
PPB ATTRIBUTE or  
FPB ATTRIBUTE

Enter the name of the attribute, as given in the table later. The utility displays the current value, then re-  
quests  
NEW VALUE.

Enter the new value required. The utility then returns to the select-function loop.

## File Attributes

Table 4-2 gives the keywords for each file attribute that can be changed by the MODIFY utility, together with allowable values for each attribute. Table 4-3 gives the keywords of each PPB attribute that can be changed, and allowable values for each.

Note that each modification is performed in turn, so that the keywords PRINT.FPB and PRINT.PPB will reflect the FPB and PPB after any modifications specified **previously** in the message to MODIFY, but before any modifications are made that are specified after the print request.

### Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
ATTRIBUTE VALUE MISSING	Value is either missing or incorrect; other modifications carried out.	Check current values by PRINT.FPB or PRINT.PPB; then use utility for the attr- ibute in error.
KEYWORD IN ERROR	Self-explanatory; other modifications carried out.	As above.
ATTRIBUTE-VAL INCONSISTENT	The attribute being assigned cannot take the value being given; other mod- ifications carried out.	As above.



MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
INCORRECT ATTRIBUTE	A value is being assigned to a value, rather than to an attribute; other modifications carried out.	As above.
DEVICE - MYUSE INCONSISTENT	Incompatible values of these file attributes. This is a warning that the program may give an error when executed.	Use MODIFY to correct either or both of these fields.
FILE-SIZE TOO LARGE	Value for FILE.SIZE is incorrect. This is a warning.	Check with PRINT.FPB if necessary and correct the attribute.
TOO MANY BUFFERS	Value for NO.BUFFERS is incorrect. This is a warning.	As above.
REC. NOT INTEGRAL OF BUF.	The buffer size is not an exact multiple of the record size. This is a warning.	Use MODIFY to correct one or both of these attributes before running program.
CODE FILE NAME IN ERROR	Self-explanatory; all modifications are ignored.	Re-input.
FILE NAME NOT FOUND	The internal file name is not in code file; all modifications are ignored.	As above.
CURRENCY SYMBOL EXPECTED	Self explanatory	Re-input.
NUMERIC ATTRIBUTE - VAL REQD	Non-numeric characters were input where a numeric value is needed.	Check with tables 4.2 and 4.3 and re-input.
FILE NOT SPECIFIED	Missing keyword "FILE"	Re-input.
PPB NOT SPECIFIED	Attempt to modify PPB while not in PPB mode.	Re-input.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
NOT AN INDEXED FILE	Attempted to use an indexed-file attribute on a non-indexed file.	Check initial input to MODIFY
NOT A COBOL PROGRAM FILE	Wrong PPB attribute used (see table)	Re-input.
NOT AN MPL PROGRAM FILE	Wrong PPB attribute used (see table)	Re-input.
CODE FILE - BAD FILE TYPE	Attempted to modify a code file which was not of type CODE.	None.
CODE FILE IN USE	The specified code file is in use.	Wait until the task using code file goes to end of job.
CANNOT OPEN THAT CODE FILE	Attempted to modify a code file which was not available for some reason.	None.
NUMBER TOO BIG	Attempted to assign a value greater than 65535.	None.
MUST HAVE 0 < KEY LENGTH < 29	Attempted to assign a KEY LENGTH value out of range.	None.

Table 4-2. File Attributes Accessible by Modify

file attribute name	allowable values
MFID	1-7 alphanumeric characters
FID	1-12 alphanumeric characters
REEL	3 decimal digits in range 000-999
DEVICE	one of the mnemonics given in Table 4-4
RECORD	1-5 decimal digits in range 0-65535
BUFFER	1-5 decimal digits in range 0-65535
FILESIZE	1-7 decimal digits in range 0-1048560
NO.BUFFERS	1-2 decimal digits in range 1-16
CYCLE	2 decimal digits in range 00-99
FORMS	ON, OFF
SET.UPDATE	ON, OFF
NO.LABEL	ON, OFF
CONDITIONAL	ON, OFF
SINGLEAREA	ON, OFF
GEN.CHECK	ON, OFF
NO.REWIND	ON, OFF
REVERSE.CHECK	ON, OFF
CLOSEMODE	LOCK, PURGE, REMOVE, RELEASE, HALF.CLOSE
CRUNCH	ON, OFF
MERGE	ON, OFF
OTHERUSE	FREE, LOCK.ACCESS, LOCKED or SHARED
MYUSE	INPUT, OUTPUT, IO
EXTEND	ON, OFF
ACCESSMODE	SEQUENTIAL, STREAM, RANDOM
GEN.NO	1-5 decimal digits in range 0-65535
LAST.ACCESS	5 decimal digits in Julian date format, YYDDD
SAVE	1-3 decimal digits in range 0-999
FILE.DEFAULT	TYPE1 thru TYPE29 (see MPL Reference Manual)
NON.STANDARD	ON, OFF
D.MFID	1-7 alphanumeric characters (indexed files only)
D.FID	1-12 alphanumeric characters (indexed files only)
ROUGH.TABLE	1-5 decimal digits in range 0-65535 (indexed files only)
KEY.LENGTH	1-2 decimal digits in range 1-28 (indexed files only)
KEY.OFFSET	1-5 decimal digits in range 0-65535 (indexed files only)

**Table 4-3. PPB Attributes Accessible by Modify**

PPB attribute name	allowable values
INTERP.PACK	1-7 alphanumeric characters
INTERP.NAME	1-12 alphanumeric characters
CLASS	A, B, C
EOJ.SUPPRESS	ON, OFF
DATA.STACK	1-5 decimal digits in range 0-65535 (MPL/BIL programs only)
CONTROL.STACK	1-5 decimal digits in range 0-65535
CURRENCY.SYMBOL	one character (COBOL/RPG programs only)

**Table 4-4. Mnemonics for Device Attribute for Modify**

mneumonics	meaning
PR	any printer
KP	keyboard printer
KD	keyboard display
KB	keyboard any output
SP	serial printer
LP	line printer
CR	any card reader
CP	any card punch
CRP	any card reader/punch
CR80	80-column card reader
CP80	80-column card punch
CRP80	80-column card reader/punch
CR96	96-column card reader
CP96	96-column card punch
CRP96	96-column card reader/punch
PTR	paper tape reader
PTP	paper tape punch
MT	magnetic tape reel or cassette
MT9	magnetic tape reel
CS	magnetic tape cassette
MT9IN	magnetic tape reel without write permit
CSIN	magnetic tape cassette without write permit
DC	any disk
DM	Any mini-disk

## PB (List Printer Backup files)

This utility allows the operator to list printer backup files. The utility has two formats.

### Format 1

PB disk-name NNNNN  
or  
PBNNNNN SAVE COPIES number ON device-type

### Format 2

PB disk-name/= COPIES number ON device type

The options can be entered in any order.

The printer backup files have the identity of "PB" as the first two characters of their name and file type value "AO" (see the MCP manual for file types).

To print a printer backup file PB00011 from the system disk:

```
PB 11
PB PB00011
```

If the option SAVE is used then the backup file will be retained after printing. If the option is not used then the file is removed from the disk after printing.

The option COPIES followed by a number specifies the number of copies to be printed, between 1 and 99. The default value of 'number' is 1.

The option ON allows the operator to specify the type of printer that will be used to print the file. If used, this option will override the device type specified when the file was created.

#### Examples:

To list a printer backup file PB00100 from a disk PRDISK on serial printer and save after printing:

```
PB PRDISK/PB00100 SAVE ON SP
```

To list 5 copies of a backup file PB00101 and remove after printing:

```
PB 101 COPIES 5
```

To list 2 copies of all the printer backup files from a disk ARDISK and remove after printing:

```
PB ARDISK/= COPIES 2
```

#### Output Format:

The first page of the output listing will contain a banner type heading. The heading will contain Time, Date, Program-name, which created printer backup file, and backup file name taken from the first record of printer backup file.

If the special forms fit is set, a field in the File Parameter Block (see MCP manual) which is set programmatically, then the heading is not printed for that file.

If the special forms bit is set, then the utility will display a message and wait on accept:

```
SPECIAL FORMS REQUIRED FOR printer backup file-name CREATED BY program name
```

The operator should enter "Y" or "N" to the ACPT message to print or not to print the file.

After printing the file requiring special forms, a message is displayed to inform the operator to reset forms:

REMOVE SPECIAL FORMS

and the utility will wait on accept. The response should be

AX mix number OK

At the end of the special forms job, the printer is closed with a lock.

Output Messages:

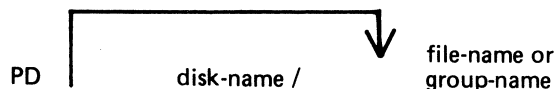
MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
INVALID FILE FORMAT IN file-name	The file specified to be printed by the utility is not a printer backup file.	None.
ABNORMAL TERMINATION FOR file-name	The PB file number found at the beginning of each data record does not match the PB file number found in the identification record of the PB file (first record).	None.

# PD (Print Disk Directory)

(a function of SYS-SUPERUTL)

This utility allows the operator to verify the presence on disk of a particular file or a group of files.

Format:



Examples:

To find out if a particular file is on disk:

```

PD PR210
PD PR2/PR020

```

To find out if a group of files is on disk:

```

PD PR2/PR0 =
PD PR3 =

```

To find out if several different files or groups are on disk:

```

PD PR3 = , PR2 =
PD GL2GL0 = , GL2/GL30 = , GL250

```

To inquire about all files on disk:

```

PD = Non valida: scave 4paris PDK =
PD PR2/=

```

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
ON LINE program-name	File found on disk if single file requested.	None.
group-name ON disk-name CONTAINS:	Group of files found on disk	None.
NOT ON LINE program-name	File not found on disk	Check input (re-input if necessary), Check for correct disk.
NO FILES FOUND IN THE FAMILY "group-name"	Group not found on disk.	Check input (re-input if necessary), Check for correct disk.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
END PD	Successful End of Job.	None.
"file-name" REQUIRES OVERFLOW DISK "disk name"	Remainder of specified file resides on another disk. PD cannot complete until appropriate disk is supplied.	Supply appropriate disk.

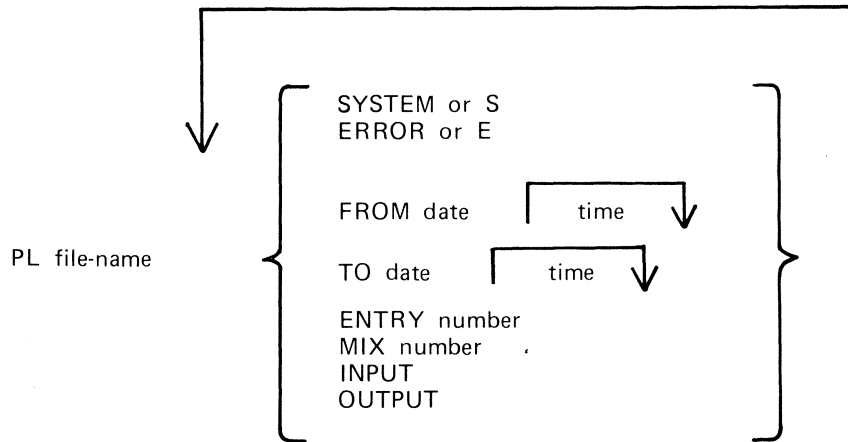
Note: See "Common Utility Output Messages" for additional aid.



## PL (Print Log Files):

This utility allows the operator to list the contents of log-files present during any one particular session.

Format:



The option "SYSTEM" (or "S") is specified to list only system messages from the log-file.

The option "ERROR" (or "E") is specified to list only error messages.

The option "FROM MM/DD/YY HH/MM/SS" is used to list the logged message from the specified date and time. If time is not specified, then 00/00/00 is assumed. MM = month, DD = day, YY = year, HH = hour, MM = minute, SS = second.

The option "TO MM/DD/YY HH/MM/SS" is used to list the messages up to that date and time. If time is not specified, then 00/00/00 is assumed.

If the option "ENTRY" is used, the utility will print starting from the record number specified by the operator.

The "MIX number" option is used to print all messages related to specified mix (number(s)).

The "INPUT" and "OUTPUT" option allows the operator to print either input or output messages.

Certain "defaults" are as follows:

- SYSTEM and ERROR messages;
- INPUT and OUTPUT messages;
- FROM 00/00/00 00:00:00;
- TO last date and time in log;
- ENTRY 1.

All entries are displayed irrespective of their mix numbers. Any of these defaults can be reset at run time. If no real-time clock was available when the file was created, then no check will be made on the "time" portion of the operator input, and "N/A" will be printed under the "TIME" heading on the report.

Entries with multiple records will only have the record number and record contents displayed; all other columns will be blank since the contents of these records will all be of the same type, and created at the same time.

Only entries which conform with either the defaults, or operator input specifications will be displayed, all others will be ignored.

The range of values for ENTRY and MIX numbers are 1-65535 and 1-254 respectively. Checks at run-time are made on the values entered and messages issued if they are in error.

18

Examples:

To print the contents of the log-file called SYS-LOG-HOLD:

PL SYS-LOG-HOLD

To print the error messages logged in the log-file called SYS-LOG-01:

PL SYS-LOG-01 ERROR

To print entries in SYS-LOG-HOLD file from record 100, related to mix number 12 from January 1, 1979 until latest date:

PL SYS-LOG-HOLD ENTRY 100 MIX 12 FROM 01/01/79

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
NO FILE NAME IN PARAM.FILE file-name	*<file-name> option was used. The utility failed to find a file-name within the first 5, 80-byte records of the file, "file-name".	Check contents of "Starfile" (use LIST) and recreate if necessary; re-enter.
NO FILE NAME FOUND IN PARAMETERS	One of the options is specified first.	Re-enter, with file-name first.
ILLEGAL OPTION text.	The option specified was not recognised.	Check input and re-enter.
ILLEGAL VALUE number.	"number" after either ENTRY or MIX is not within the allowable range of numbers.	Check input and re-enter.
ILLEGAL NUMBER OF RECORDS IN FILE filename	*<filename> option was used. "filename" file has more than 5, 80-byte records.	Check contents of "Starfile" (use LIST) and recreate if necessary.
ILLEGAL PARAMETER text	Input after a special entry is incorrect, or inappropriate, that is, "ENTRY number". If "number" was not recognizable, this would produce the error).	Check input and re-enter.
NO TIME SPECIFICATION AFTER time option	Input after either "TO" or "FROM" is incorrect or inappropriate.	Check input and re-enter.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
NO text NUMBER GIVEN	A number after a special entry (that is, after MIX) was expected but not found).	Check input and re-enter.
option OPTION ALREADY SET	One part of the operator input message contradicts the other (that is, "INPUT" and "OUTPUT" specified simultaneously).	Check input and re-enter.
ILLEGAL TIME SPECIFICATION time specification	"time specification" format incorrect (that is, "time" was typed before "date" if both are used).	Check input and re-enter.
UNABLE TO OPEN FILE	PL unable to use a required file. (EX: in use by some other program).	Verify (with PD) that file is on disk; if file is on correct disk, wait until program using file goes to EOJ and then retry PL.
NO ENTRIES FROM ENTRY NUMBER number	"ENTRY" option was used and nothing equivalent to specified "number" was found in file. Note: This option will start the processing at the first value it finds which is the same as that given. If the "match" fails, this message displayed.	Check input and re-enter.
filename IS NOT A LOG FILE	Specified file is not of "log-file" format.	Check input and re-enter if necessary.
ILLEGAL FILE NAME filename	"filename" does not conform to standard CMS format (that is, disk name might be too long).	Check input and re-enter if necessary.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
NO ENTRIES FOUND WITHIN PARAMETERS	During processing, End of File is reached and no entries have been printed.	Check input and re-enter if necessary.
PARITY ERROR IN READ OF FILE filename	(Record number at which error occurred prints immediately after message). Program encountered error on trying to read specified file. WARNING: Program normally attempts to continue processing. However, if ENTRY option has been used and error occurs while processing this command, the results might not be those requested.	
UTILITY LIMIT REACHED	This is displayed if, when analysing maintenance entry, certain information is longer than the utility can handle.	None.
UNRECOGNISED DEVICE IN ERROR ENTRY	This is displayed if, when analysing the error descriptor, no device can be found in the Hardware Configuration table which corresponds to this device type - hence no error counts can be updated from this information.	None.

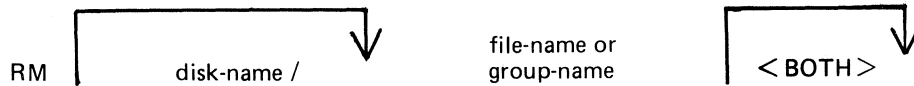
## RM (Remove Files from Disk)

(a function of SYS-SUPERUTL)

This utility allows the removal of individual files and groups of files from disk. The disk areas associated with those files are returned to the available table.

If the utility detects that a keyfile is to be removed and the <BOTH> option has been specified, then it will remove both the keyfile and the associated data file if both are on disk. If <BOTH> is not specified then only the keyfile will be removed.

Format:



Examples:

To remove a single file:

```

RM AR300
RM PR1/PR300
  
```

To remove a group of files:

```

RM AR=
RM INDISK2/IN3=
  
```

To remove several different groups and/or individual files:

```

RM IC230, IN076, INDISK1/IN2=
  
```

To remove a keyfile and associated data file:

```

RM PR200K <BOTH>
  
```

A request for the removal of a system file will cause the utility to output the following:

```

file-name IS A SYSTEM FILE
AX "mix number"/RM ACPT
  
```

Then, to remove a system file:

```

AX mix-number/RM file-name OK (mix-number is the mix number of RM).
  
```

If the operator types any other sequence the system file will not be removed.

Example:

```

RM NDL=
NDL.INTERP IS A SYSTEM FILE
12/RM ACPT
AX 12/RM NDL.INTERP OK
  
```

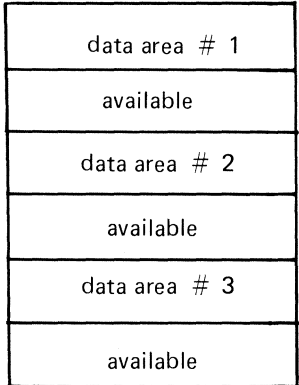
Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
"filename" REMOVED	File was removed.	None.
"filename" NOT REMOVED NOT FOUND	Specified file was not removed.	Check input (re-input if necessary); Check for correct disk.
"filename" NOT REMOVED - IN USE	Specified file was not removed because it is currently being used by the system.	wait until file is no longer in use, then re-input.
"filename" NOT REMOVED - SYSTEM FILE	Specified file was not removed because it is a "system file" (for example, MCP, an interpreter, etc).	If file is to be removed, type "AX mix-number/RM file-name OK". If file is not to be removed, type AX mix/RM NO
INDEXED PAIR "file name" "file-name" REMOVED	Keyfile and associated data file were removed.	None.
INDEXED PAIR "filename" "filename" NOT REMOVED	Keyfile and associated data file were not removed. This message is followed by the reason.	Check input (re-input if necessary) or Check for correct disk.
INDEXED PAIR "filename" "filename" NOT REMOVED - IN USE	Specified keyfile and data file were not removed because at least one is currently being used by the system.	Wait until files are no longer in use, then re-input.

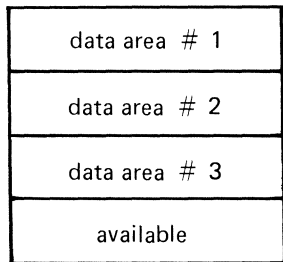
*Computer & all*

## SQ (Squash Disk)

When a disk unit is used extensively with a high degree of file activity involving creation and removal of files then it is possible for the available space on the disk to become so fragmented that it is increasingly difficult to find enough space in one single area to satisfy requests for disk space. This results in a degradation of system throughput with an increasing incidence of "NO USER DISK" failures and extra time needed to search through available areas. This situation is known as "checkerboarding" of the disk. In the extreme case each area of disk in use is separated by an available area, as shown in the diagram below:



The SQ utility is designed to eliminate checkerboarding of disk, either for the whole disk or part of the disk. This process is called "squashing" disk and is accomplished by moving each data area in turn to the first available area at a lower address. If an entire disk is squashed then all available areas are merged into one area at high-address end of the disk, as in the next diagram:



The options available within the SQ utility are:

### Squash of a complete disk.

All data areas are moved to successively lower addresses until only one available area is left (as in diagram above).

### Partial squash

Only data areas within a default section of the disk are moved to lower addresses within the section.

### Fast squash

The aim of a fast squash is to create an available area of disk of a requested size. Only those data areas are moved which will allow an available area of a sufficient size to be created.

### Economic squash

In this case, data areas are only moved if the gain in terms of available space justifies the time spent in movement of the data area. As an example, consider the following case:

24

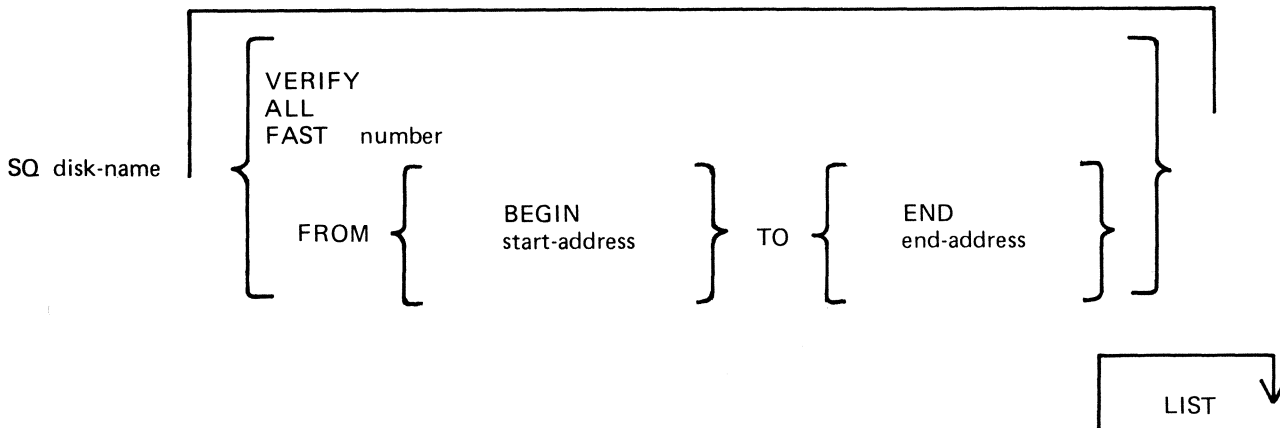
data area # 1
available # 1
data area # 2
available # 2

where data area #1 is 100 units, data area #2 is 200 units and both available areas are 1 unit each. If available areas are merged the available area gained would be 2 units. However, to acquire these 2 units, the 200 units of data area # 2 would have to be moved. Therefore an "economic squash" would not move data area #2. In general terms, an economic squash will ignore small available areas that are interspersed in large data areas. However, in some cases an economic squash will have the same effects as a full squash.

With all options of SQ a further option is available to print a map of the entire disk in disk-address order both before and after squashing action.

Input is as follows:

Format:



note: the number is in the range 1 to 65535; the start-address and end-address are 6-digit hexadecimal disk addresses, for example, 000AB3, 01A375.

Examples:

To perform an economic squash of disk PR2:

SQ PR2

To check the integrity of disk PR2:

SQ PR2 VERIFY

To perform a full squash of disk PR2 and list the disk map:

SQ PR2 ALL LIST

To move data areas to provide one available area of 1000 sectors on disk PR2:

SQ PR2 FAST 1000

To perform a partial squash on sectors 0 through 512 of disk PR2:

SQ PR2 FROM BEGIN TO 000200



25

To perform a partial squash on sectors 512 through 4096 of disk PR2:  
AQ PR2 FROM 000200 to 001000

To perform a partial squash on sectors 4096 to the last addressable sector of disk PR2:  
SQ PR2 FROM 001000 TO END

Before performing any function which involves physically moving data areas, the integrity of the disk is checked. Integrity checking involves analyzing disk assignment to verify that the entire area of the disk is described in the file directories and available table, checking the directories themselves and attempting to resolve anomalies (for example, missing areas or overlapping areas). Only after the integrity is verified are areas of disk physically moved.

Certain areas of disk will not be moved in any circumstance. These are areas of disk currently marked as in use, and any system log files. In addition, SQ can only be run in a suitable mix, as defined by the MCP to safeguard the integrity of the disk. No user program can be run with SQ. During execution of SQ the MCP will reject any attempt to execute any utility or user program.

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
LARGEST AVAILABLE AREA IS number SECTORS. TOTAL AVAILABLE IS number SECTORS IN number AREA(s). ***SQ COMPLETED***	Given on successful completion of SQ.	None.
NON-FILE DIRECTORY FULL - "PARTIAL" SQUASH REQUIRED	Display during verification phase if there are no free entries when attempting to add entries to the available table if missing areas are detected.	Run SQ with VERIFY LIST options. Examine the disk map to discover a section of the disk that can be squashed to create free entries in the available table Example: The pattern File Area Available File Area Available if squashed will create one free entry in the available table. In general the # of free entries to be created equals the # of missing areas that are not contiguous with any available area.
SQ INVALID - NO INITIATING MESSAGE	Self-explanatory	Re-input.

28

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
SQ INVALID - INVALID PACK-ID	Pack-id specified is longer than 7 characters.	Re-input.
SQ INVALID - INVALID SYNTAX	Self-explanatory	Re-input.
SQ INVALID - INTEGER MUST NOT BE GREATER THAN 65535	The maximum # of sectors requested for the FAST option is 65535.	Re-input.
SQ INVALID - INVALID ADDRESS	No address or an invalid address (for example, 004G15) was specified for the FROM or TO address with partial squash.	Re-input.
*** BAD SECTOR AND file-name OVERLAP - NO WAY TO SEPARATE THEM. SAVE AND/OR PURGE AND AT LEAST RETURN SQ VERIFY	Area marked in available table as "bad" overlaps with area allocated to file.	SQ cannot resolve this. Integrity of file is suspect. Remove the file after copying it to another disk for examination if necessary and run SQ VERIFY.
*** file-name AND file-name OVERLAP - THERE IS NO WAY TO SEPARATE THEM. SAVE ONE OR BOTH, PURGE AND AT LEAST RERUN SQ VERIFY	Area allocated to file overlaps with area allocated to another file.	The two areas cannot be separated. Copy each file individually to another disk for later examination if required, remove them both, rerun SQ VERIFY
SQ ABORTED - REQUESTED AREA ALREADY EXISTS	Request was made with FAST option for an available area which already exists.	None.
*** SQ ABORTED - INVALID DISK ALLOC. UNIT = 0	Disk label is probably corrupted.	Disk must be assumed useless and should be re-initialized.
*** DIRECTORY FID NEQ HEADER FID FOR FILE file-name. CORRECT USING CH AND RESTART SQ	Name of file in file directory name list does not match disk file header. The file-name displayed is that in the name list.	Enter "CH <FILE-ID> TO <FILE-ID> to correct the anomaly (this rewrites the disk file header).

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
SQ INVALID - SPECIFIED DISK NOT AVAILABLE	Self-explanatory.	Make disk present and rerun SQ; check input.
*** SQ ABORTED NOTHING TO SQUASH IN THAT AREA	A partial squash was requested and SQ found nothing to do in that section.	None.
SQ ABORTED - NO WAY TO GET REQUIRED AREA	Area size specified in the FAST option cannot be obtained either because it is larger than the total available space or because certain areas cannot be moved to release available space. For example: Area # 1 100 units Available # 1 1000 units Area # 2 100 units Available # 1 cannot be used if areas # 1 and # 2 are in-use or system log files because areas # 1 and # 2 cannot then be moved.	Attempt to remove unwanted user files: re-input SQ.
*** MEMORY INCONS- ISTENCY OR SOME OTHER IRRECOVERABLE PROBLEMS - RERUN SQ	Internal work-tables in memory (used by S6) are corrupted.	Rerun SQ VERIFY. If problem persists, request technical assistance.
*** ADDRESS MISMATCH - SAVE AND REINITIATE THE DISK	Some addresses in disk directory are probably corrupted.	Try to dump or copy files from the disk. Disk must be reinit- ialized before re-use.
*** IRRECOVERABLE ERROR ON DISK - SAVE AND/OR REINITIALIZE	Disk is corrupt.	Try to dump or copy files from the disk. Disk must be reinit- ialized before re-use.
*** TOO MANY FILES OPEN AND/OR BAD AREAS - NO WAY TO SQUASH THE DISK	Self-explanatory.	Save required files from disk, then reinitialize the disk.

28

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
*** DISK INTEGRITY SUSPECT - USAGE MAP OF THE DISK WILL BE PRINTED (IF LIST IS NOT SET)	SQ has detected and resolved an overlap situation; but disk remains suspect.	Scrutinise disk map, request technical assistance.
*** AREAS STILL MISSING - RERUN SQ VERIFY	A FAST squash has detected that areas of disk are not accounted for.	SQ VERIFY will return missing areas to the available table.
*** LAST SQ EXECUTION WAS ABNORMALLY TERMINATED WHEN MOVING FILE - file-name-INTEGRITY OF FILE SUSPECT. EXECUTION CONTINUES	System crash occurred while file areas was being moved.	Remove suspect file.
*** SQ ABORTED - I/O ERROR AT DISK ADDRESS @NNNNNN@	Hard error on disk persists after 10 retries.	If address is outside directory area, remove offending area with XD utility.
*** I/O ERROR ON file-name FILE SKIPPED - EXECUTION CONTINUES.	Hard disk error encountered. File is not moved.	None.
*** WRITE ERROR IN NEW LOCATION WHEN MOVING FILE AREA. DISK ADDRESS - @NNNNNN@. THIS AREA SKIPPED - EXECUTION CONTINUING	Hard error encountered when moving file area to available area.	Available area is left as available and should be XD-ed after squash EOJ.

**NOTE**

Error messages marked with “\*\*\*” indicate that a hardware or system software error has occurred or that the disk itself is suspect. If these persist, request technical assistance.

**General Guidelines**

If the information contained on a disk is important always ensure that backup exists before attempting to squash it.

Always run “SQ VERIFY” before running an actual squash. This will give an indication of the state of the disk.

Do not allow disks to become too fragmented before squashing them. A full squash can be a lengthy process and can be avoided by running “SQ VERIFY” on a regular basis and running partial squash when the disk starts checker-boarding.

“SQ VERIFY” is a means of checking the integrity of any disk and if run on a regular basis may help pinpoint sooner rather than later any degradation in hardware performance or system software bugs. For disks that are in constant use “SQ VERIFY” should be run immediately after the first clear start of the day. This can help prevent catastrophic losses of information.

## SYCOPY (Copy Library Tapes)

This utility allows the operator to copy, compare and merge library tapes and cassettes (tapes created by DUMP/UNLOAD utilities).

Format:

```
SYCOPY { CPY   - number - file-equate
         or
         CMP   - number - file-equate
         or
         CCM   - number - file-equate
         or
         MRG   - file-equate
         or
         TEACH
```

Where number is in the range 1 to 7 and the default value is 1.

file-equate is defined as one or more of the following entries:

FI default-name NAME actual-name

The default-names for input tape is "IN" and for output tapes "TAPE.1" through "TAPE.7", except in MRG function where the input tape names are "SYSTEM" and "UPDATE" and the output tape name is "OUT". All output tapes are locked after job termination.

The copy function, CPY, allows for one input tape to be copied to up to seven output tapes.

Examples:

To copy one input tape labelled "IN" to two output tapes, labelled "TAPE.1" and "TAPE.2":

```
SYCOPY CPY 2
```

To copy one input tape labelled "FRED" to one output tape, labelled "FRED.OUT":

```
SYCOPY CPY FI IN NAME FRED FI TAPE.1 NAME FRED.OUT
```

The compare function, CPM, allows for one input tape to be compared to up to seven input tapes.

Examples:

To compare one tape named "IN" with three tapes named "TAPE.1", "TAPE.2" and "TAPE.3":

```
SYCOPY CPM 3
```

To compare one tape named "FRED" with one tape named "TAPE.1":

```
SYCOPY CPM1FI IN NAME FRED
```

The copy and compare function, CCM, allows the copying of one input tape to up to seven output tapes and then the comparison of the same input tape with the output tapes.

Examples:

To copy one input tape named "IN" to three output tapes named "TAPE.1", "TAPE.2", "TAPE.3" and then to compare "IN" with "TAPE.1", "TAPE.2" and "TAPE.3":

```
SYCOPY CCM 3
```

The merge function, MRG, allows two input tapes to be merged to one output tape. If there are duplicate files on the input tapes then the version on the second tape (default name UPDATE) will be copied to the output tape.

Examples:

To merge two input tapes named "SYSTEM" and "UPDATE" and output one tape named "OUT":  
 SYCOPY MRG

To merge two input tapes labelled "A" and "UPDATE" to one output tape named "B":  
 SYCOPY MRG FI SYSTEM NAME A FI OUT NAME B

The TEACH function allows the operator to list the syntax of the utility on a printer.

Example:

SYCOPY TEACH

Output Messages:

MESSAGE	POSSIBLE CAUSE	SUGGESTED ACTION
number TAPE(S) COPIED	successful copy function	none
number TAPE(S) COMPARED	successful compare function	none
number TAPE(S) COPIED AND COMPARED	successful copy and compare function	none
2 TAPE(S) MERGED	successful merge function	none
INVALID SYNTAX - SYCOPY ABORTED	The initiating message contains some invalid specification, for example: invalid or mis-spelled option; number following MRG; file equate to an invalid file-name; mis-spelled file- equate clause	Check input and re-enter if necessary
COMPARISON ERROR ON tape-name - TAPES NOT IDENTICAL - SYCOPY ABORTED	The compare function failed, possibly due to a hardware mal- function	Re-run the SYCOPY function with good tapes and drive
RECORD SIZE DIFFERENT ON tape-name	The record size of the tape specified differs from the record size of the input tape	Check input tape and re-run SYCOPY if necessary
WAITING tape-name/ FILE000 (or FI00000) AT NO FILE	Specified tape is not available, or tape drive cannot read tape correctly	Mount the correct tape or investigate tape drive for errors

## TAPELR (List Library Tape Directory)

This utility allows the operator to print detailed information about the library tape files. Output will appear either on the line printer or the console printer.

Tapes about which information is required are identified by "library-tape-name". More than one tape name may be requested during a single run of TAPELR.

Format:

TAPELR library-tape-name

Examples:

To print detailed information about the files on a tape called PRTAPE:

TAPELR PRTAPE

To print detailed information about the files on tapes called PRTAPE and ICTAPE:

TAPELR PRTAPE ICTAPE

Output format:

Eight columns of information will appear for each library tape indicated. The column headings, the format of the "values" these columns contain, and the significance of these "values" is as follows:

HEADING	VALUE	SIGNIFICANCE
-----	-----	-----
FILE NAME	12 character	File name
ACTUAL SIZE	7 digits	Number of records in this file
MAXIMUM SIZE	7 digits	Maximum # of records this file may contain.
RECORD SIZE	5 digits	# of characters in each record
RECS/BLOCK	5 digits	# of records in each block
CREATED	5 digits	Date file was created (Julian YYDDD)
ACCESSED	5 digits	Date file was last accessed by a program (Julian YYDDD)
FILE TYPE	8 characters	See Note below

Note: FILE TYPE will be one of the following:

- DATA - normal data file
- CODE - object program file
- KEY - key file
- SYSTEM - system file (for example, MCP, interpreters)
- SRCELANG - source language file
- SRCELIBR - source library file

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
Library-tape-name NOT A RECOGNIZED DUMP TAPE	This tape was not created by either DUMP or UNLOAD functions of LD utility. It is ignored by the TAPELR utility.	None.

Note: Refer to "Common Utility Output Messages" for additional messages.



NV

## TAPEPD (Print Name of a Library Tape)

This utility allows the operator to print the names of files found on a library-tape. More than one tape name may be requested during a single run of TAPEPD.

Format:

**TAPEPD library-tape-name**

Examples:

To print the names of files found on a tape called PRTAPE:

**TAPEPD PRTAPE**

To print the names of the files found on tapes called PRTAPE, ICTAPE, and GLTAPE:

**TAPEPD PRTAPE ICTAPE GLTAPE**

Output format:

For each tape requested, the following information is displayed:

MT library-tape-name DUMPED ON day of week DD month YY contains:

This message precedes the names of files found on each tape. The list itself contains 3 files per line.

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
library-tape-name NOT A RECOGNIZED DUMP TAPE	This tape was not created by either the DUMP or UNLOAD function of LD utility. It is ignored by the TAPEPD utility.	None.
<b>END TAPEPD</b>	<b>End of job message.</b>	<b>None.</b>

Note: Refer to "Common Utility Output Messages" for additional messages.

## TL (Transfer Log Files)

If logging is enabled following any warmstart, a number of "log-files" are in use. The purpose of these files is to maintain a record of all input/output messages that appear on the SPO within this given period of time.

In order to produce easy access to all the files, they are consolidated into one large file. This is done through the use of the TL utility.

Format:



"File-name" is the name the user wishes the consolidated file to be called.

In order to permit a correct consolidation, there may be NO other programs running at the time when TL is initiated.

The utility will determine the number of files to be consolidated and also the size of the consolidated file. It will then transfer each "ready-to-transfer", closing the consolidated file after each log-file has been transferred, until it reaches the file which was in an "active" state at time of execution of TL.

If the "RECOVER" option has been specified, then the utility will enter the "active" log file when it reaches it and it is not in use, and transfer all entries up to the latest. If the "active" file is in use then the utility displays "ILLEGAL USE OF RECOVER PARAMETER, ACTIVE FILE NOT CONSOLIDATED", and will consolidate only the "ready-to-transfer" files.

If the "RECOVER" option has not been specified, consolidation will **end when** the "active" file is reached.

All log-files transferred will be left in a "transferred" state.

### Examples:

To transfer all "ready-to-transfer" SYS-LOG files to LOGHOLD:

```
TL LOGHOLD
```

To transfer all "ready-to-transfer" SYS-LOG files to LOGHOLD on the disk called ARDISK1:

```
TL ARDISK1/LOGHOLD
```

### Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
ILLEGAL FILENAME file-name	Specified file-name contains too many characters.	Check input and re-input.
UNABLE TO OPEN TRANSFER FILE file-name	Utility cannot "create" a consolidated file with the name requested (for example, no disk space, no disk media).	Check with KA on available space: use RM if necessary and re-input. Check drive media.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
FILE file-name NOT FOUND	The utility was unable to find the specified file-name	Correct the file-name and re-input.
FILE file-name IN USE	The specified file is in use.	Wait until the task using the specified file goes to end of job.
NO LOG FILES FOUND FOR CONSOLIDATION	This is displayed if the utility, on attempting to determine the number and sizes of log files to be transferred, was unable to find any with a file-id of the form "SYS-LOG-NN"	None.
ILLEGAL USE OF RECOVER PARAMETER - ACTIVE FILE NOT CONSOLIDATED	The utility is executed with RECOVER option. The active file can only be consolidated at warmstart.	None.
INVALID CHARACTER IN IDENTIFIER input	Some invalid character has been typed in the input.	Correct the input and re-input.
TRANSFER COMPLETED	Successful termination of the utility.	None.
NO READY-TO-TRANSFER FILES FOUND	No log-files in either an "active" or "ready-to-transfer" state were found. No consolidation will occur.	None.
NO ACTIVE LOG FILE FOUND	No file in the "active" state was found.	None
TRANSFER COMPLETED	IL successful End of Job.	None.
PARITY ERROR IN READ OF FILE file-name or PARITY ERROR ON WRITE TO FILE file-name	Read or write parity errors found for specified files. The record number at which error occurred will be displayed.	None

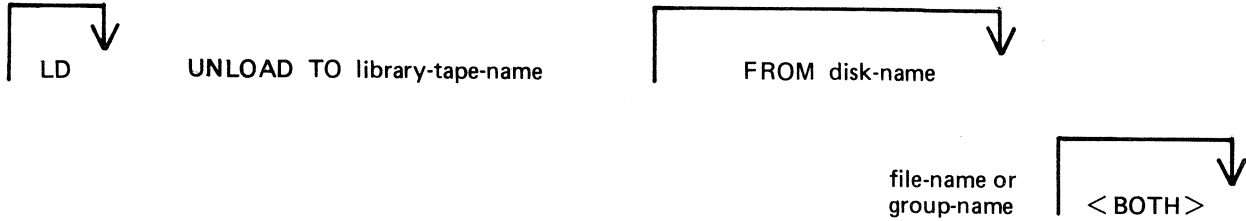
37

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
	If utility continues processing, then the message "CONTINUING PROCESS" will display. Otherwise TL will go to End of Job and leave partially consolidated file, depending on where error occurred.	
NO FILE NAME FOUND IN PARAMETERS	The "RECOVER" option was used and a file name was not given.	Correct the input and re-enter.
CONTINUE PROCESS	This message followed the parity error message if utility able to continue process.	None.

## UNLOAD (Unload Files from Disk to Library Tape)

This function, a part of the utility LD, allows the operator to copy files from disk to a library tape. The files will be deleted from the disk after they have been copied to the tape.

Format:



If the <BOTH> option is used immediately after a request to dump a keyfile, the associated data file will also be dumped, provided it resides on disk.

Examples:

To dump all files from disk beginning with the letters, "PR" to a tape called PRTAPE; and then remove them from disk:

```
UNLOAD TO PRTAPE PR=
```

To dump the keyfile called AR200K and its data file from a disk called APBU to a tape called ARTAPE, removing them from disk after dumping:

```
UNLOAD TO ARTAPE FROM APBU AP200K <BOTH>
```

To dump from the system disk files called AP020, AP030, and APTASK to a tape called ARTAPE; removing them from disk after dumping:

```
LD UNLOAD TO ARTAPE AP020 AP030 APTASK
```

Since "UNLOAD" is a part of the utility LD, "LD" is actually what will appear in a mix message. To discontinue the UNLOAD function, DS mix-number/LD must be used.

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
NO FILES IN THE FAMILY group-name ON DISK disk-name FOR UNLOAD	Specified group was not found on this disk.	Check input; re-input if necessary. Check for correct disk.
NO FILE file-name ON DISK disk-name FOR UNLOAD	Specified file was not found on this disk.	Check input and re-input if necessary. Check for correct disk.
file-name NOT DUMPED - IN OUTPUT USE DUMP ABANDONED - TAPE BEING PURGED	Specified file cannot be dumped as it is in use. If "ABANDONED" message is given, tape is purged and UNLOAD goes to EOJ.	Wait until the file is not in use and then re-enter the UNLOAD message for all files.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
file-name NOT DUMPED - HAS BEEN REMOVED. DUMP ABANDONED - TAPE BEING PURGED	Specified file was removed between the start of UNLOAD and the time when it was to be dumped to tape. File cannot be dumped. Tape is purged an UNLOAD goes to EOJ.	None.
file-name NOT DUMPED - HAS BEEN ALTERED> DUMP ABANDONED - TAPE BEING PURGED.	Contents of specified file were changed between the start of UNLOAD and the time when it was to be dumped to tape. File cannot be dumped.	Check input and re-enter if necessary.
file-name LOAD/DUMP DISCREPANCY	End of file has been reached before expected. Implies erroneous Disk File Header.	
file-name NOT DUMPED - DATA FILE NOT ON LINE	<BOTH> option was specified, but data file was not found on disk.	If specified data file dump is required, supply utility with backup copy of file if exists.
DUPLICATE file-name ALREADY BEING DUMPED	More than one request was made to UNLOAD same file.	None.
file-name REMOVED	UNLOAD successful; original file on disk was removed.	None.
file-name DUMPED	UNLOAD successful.	

Note: Refer to "Common Utility Output Messages" for additional messages.



The "number" cannot be less than the last record obtained from the old file, or greater than the number of records in the file. During the process of locating the required record, all records from and including the last record processed, up to the one immediately prior to the selected record, will be copied from the existing file to the new file. When found the selected record will be printed, with its record number in the old file followed by the record number that the next record written to the new file will take. "Record Modify" (PK2) or "Record Insert" (PK4) may then be selected. Note that a record inserted by Record Insert mode will be positioned after the selected record in the new file. Selecting Record "0" allows records to be inserted before Record 1 of the old file.

PK2 is used to make alterations to existing records. This PK operates as PK2 in the CREATE utility (see CREATE for details).

PK4 allows the operator to insert additional records in the new file after the last selected record of the old file. Input may be made in accordance with the specified tab stops. The utility prints the record number in the old file of the last record taken from the old file, and the record number in the new file, of the next record to be output, prior to accepting keyboard input. When all insertions have been made at a particular point in the file, an available PK may be pressed to select the next mode or terminate the utility. NOTE: to insert a record at the beginning of the new file, Record "0" should be selected in Record Select Mode, prior to Selecting Record Insert Mode.

Examples:

To update a source file called "APFILE" of record size 40 bytes into a file called "APFILE2".

```
UPDATE APFILE 5 10 15 20 TO APFILE2
```

The utility will illuminate PK1 and PK6. By pressing PK1, next sequential record will be selected and printed.

As the utility is already in the Record Select Mode, by typing a record number, the specified record number and its contents are printed.

```
4 4 ABCDEFGHIJKLMNOPQRST
```

Note that the first "4" is the sequence number in the old "APFILE" and the second "4" is the sequence number in the "APFILE2" file.

At this point the following PKs are available for selection:

- PK1 - select next sequential record and print
- PK2 - modify the selected record
- PK4 - insert new record after selected record (that is, "4")
- PK5 - delete the last selected record by selecting next record
- PK6 - terminate the utility

To replace characters within a selected record, press PK2 and type the replacement

```
D:ZZZZ: OCK1
```

resulting in

```
4 4 ABCD'ZZZZ'JKLMNOPQRST
```

To insert characters within a selected record, type

```
Z:XXXXXX: OCK2
```

resulting in

```
4 4 ABCDZXXXXXXXXZZZOPQRST
```



10

To insert a record after record 7 of the existing file, press PK3 (Record Select Mode) and type a record number.

7 OCK1

Note: At this point the record selection number given cannot be less than the last selected record, for example, records from 1 through 3 cannot be selected).

Press PK4 (Record Insert Mode) and utility will print last selected record number on left and next record number after that and allows operator to key in record to be inserted.

7 8 AAAAAA

The record inserted will have a sequence number of "8" in the file "APFILE" and will contain "AAAAAA".

**Output messages:**

Refer to the section on the "CREATE" utility for output messages.

*AW*

32

## WL (What Log file)

This utility allows the operator to determine the number of log files present and their status.

Format:

WL

The utility displays the following information:

FILE STATE	SYS-LOG-
ACTIVE	nn
READY-TO-TRANSFER	nn nn etc
TRANSFERRED	nn nn etc
NEXT-ACTIVE	nn (nn)
NOT USED	nn nn etc

where nn represents values between 01 and 16.

When the ACTIVE file becomes full, the MCP executes the WL utility and hence the operator is informed of the states of log files. The user can then decide whether or not to execute TL so as to preserve the log files before any overwriting of log files occurs.

Output messages:

MESSAGE	POSSIBLE CAUSE	SUGGESTED ACTION
NO SYS-LOG-NN FILES FOUND	Logging has not been initiated	Request technical assistance if logging is required (refer to appropriate section of the SOG)

*è una informazione che va nella Directory del disco ed elimina i settori corrotti*

## **XD (Delete Bad Disk Sectors)**

This utility allows the disk directory to be marked such that selected portions of the disk will not be used. The utility will normally be used after recurrent errors of the message

DK...ERROR

where the dots indicate further information. Refer to section 7, MCP output messages, for the following numbered messages:

- 2 PARITY ERROR
- 3 TIMEOUT ERROR
- 4 ADDRESS ERROR
- 45 PARITY ERROR (fatal to program)
- 46 TIMEOUT ERROR (fatal to program)
- 47 ADDRESS ERROR (fatal to program)

The further information will indicate the disk address at which the failure occurred

The utility is initiated as follows:

Format:

**XD disk-name address length**

The disk-name is the disk-id of the disk from which sectors are to be deleted. The area to be deleted is given in hexadecimal by the starting address and length.

Example:

To delete sixty-four sectors starting from hex 395F from disk PR2B:

XD PR2B 395F 40

*in esadecimale in esadecimale*

### **NOTE**

The specified sectors must not be in use as part of a file. The area must be made available by first removing any file if necessary.

Warnings:

Once sectors are deleted via XD from a disk, they can be restored to use only by a disk initialization. Do not therefore XD a larger area than required.

As XD alters the disk directory, do not run any other programs with it.

Do not execute XD from the same disk as the one from which sectors are to be deleted: for example, it is recommended that XD is always executed from the system disk and always deletes sectors from a user disk.

Output messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
@length@ SECTORS FROM @address@ DELETED	Successful termination of XD.	None.
DISK disk-name FOR XD NOT AVAILABLE	Specified disk is not available.	Check input: make disk ready.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
ONLY @length@ SECTORS CAN BE DETECTED	Only the given length can be XD-ed because the amount specified is greater than the available space at that point.	Use the KA utility yo determine if any files can be removed to increase the available space surrounding the bad area.
AVAILABLE TABLE FULL - ENTRY @address@ @length@ LOST	No entries left in available space table for XD to complete properly.	The disk may still be used, but a KA will indicate some sectors which cannot be access- ed; these may only be retrieved by initial- izing the disk.
SECTORS FOR XD NOT AVAILABLE	Requested sectors are allocated to a file, missing, or previously XD-ed.	Refer to a KA list- ing to determine which file, if any, can be RM-ed to make the sectors available.
CANNOT DELETE SECTORS FROM @address@	Only the part of the area specified for deletion is available. The utility will remove the available area only.	None
ADDRESS @address@ BEYOND END OF DISK disk-name	The sector address specified is greater than the address of the last physical sector on the disk.	None

Note: Refer to "Common Utility Output Messages" for additional messages.

# SECTION 5

## THE SORT/MERGE

### INTRODUCTION

This section describes the capabilities of the SORT facility. There are two modules: the sort itself, known as the "sort intrinsic" (file-name SORTINTRINS), and an interface to this intrinsic which allows the user to specify particular sorts and merges. The latter module is sometimes called "the sort", but is more properly called the "sort language processor" (file name SORT). The sort intrinsic is implementation-dependent, as it uses specific hardware features where possible (although output messages are standardized), while the sort language processor is a CMS common item.

This section first describes the user interface to the sort, and then covers the various facilities in some detail.

The interface to the sort from COBOL programs is described in the COBOL language reference manual.

### GENERAL FEATURES

The following capabilities are provided :

The records within a file may be sorted on a series of specified keys, each key ascending or descending, using a regular sort or an in-place sort.

A tagfile (suitable for use as an ADDROUT file in RPG or for limited access in COBOL) may be created from a file using a series of keys, each ascending or descending.

A key file (suitable for full indexed access) may be created using a specified unsigned key (ascending only), with an optional check for duplicate keys.

A number of files may be merged using a series of keys, each ascending or descending.

## INVOKING THE SORT

The sort is executed by entering the name SORT preceded by disk-name if not on the system disk, and followed by the sort language specifications or an asterisk plus "star-file" name. The "star-file" contains the sort-language statements, and may reside on card, cassette, or disk. The star-file name may be omitted: in which case the sort statements must be on a system disk file named SORTSPEC.

### Examples:

To invoke the sort using a star-file named SRTLANG on the system disk:

```
SORT*SRRTLANG
```

To execute the sort from disk PB4 using a star-file SORTSPEC on the same disk:

```
PB4/SORT*PB4/SORTSPEC
```

Note that the star-file must have a record size of not more than 90 bytes. If the sort specification is given in the initiating message it cannot be longer than 255 characters. If the sort statement is zipped from a user program it cannot be longer than 716 characters. If it is not possible to specify a complex sort or merge within these limitations, a star-file should be used.

For a one-part star-file name of 7 characters or less, the file will be searched for first on cards, then on cassette, then on the system disk. For a two-part name the file will be searched for on a user disk. For a one-part name of more than 7 characters, the file will be searched for on the system disk.

If the required file is not found, the sort displays

```
FILE filename UNAVAILABLE  
FIX AND REPLY "OK" ELSE <NULL>
```

and waits on an ACCEPT.

There are two alternative responses:

make the file present and enter OK to the ACCEPT, to resume execution, or  
enter a null response (terminator only) to the ACCEPT, to cause EOJ.

If the specification statements are provided in the initiating message, control characters such as carriage return and line feed are treated as space characters.

A star-file on cassette must be created by the COPY utility, not the LD utility.

A star-file on disk must be of type data or source, and should not be in use by other programs.

Input statements may be printed on the printer, unless inhibited by a user option (see later), or if provided in the initiating message.

*sort* → *archive*  
*sort* → *create Keyfile*

## THE SORT LANGUAGE

The specification for a sort consists of three statements:

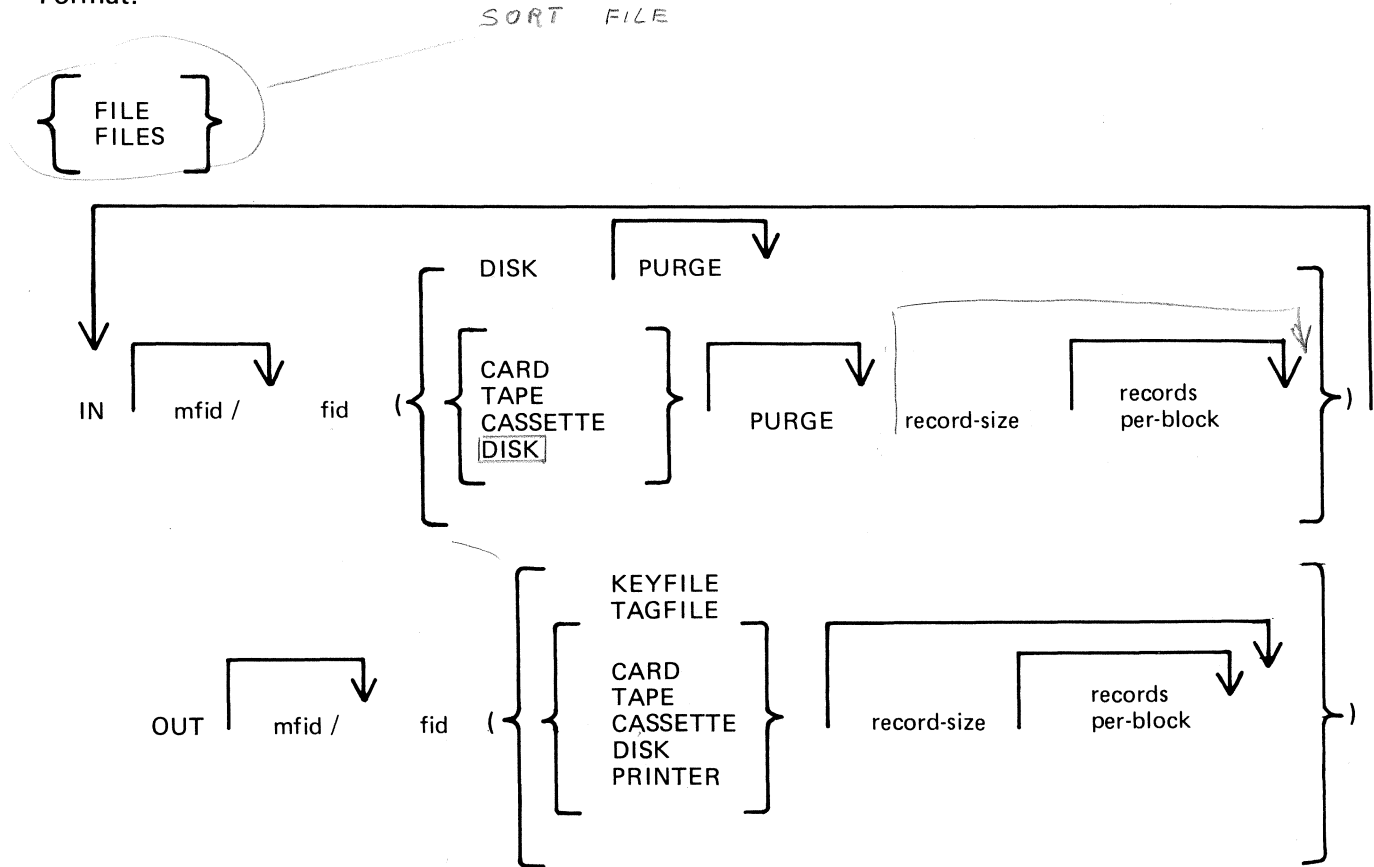
- ① the FILE statement
- ② the KEY statement
- ③ the USER-OPTION statement

There must be one file statement, one key statement, and, optionally, one or more user-option statements, in any particular sort invocation. All keywords are reserved: that is, they can only be used in the place specified below and cannot be used for other purposes such as filename.

### The File Statement

This consists of two parts; the first describes the input file(s) and the second describes the output file. Multiple input files are used only for the merge, which is specified as a user-option (see later). A sort must have only one input file; a merge may have up to 16 input files.

Format:



The parentheses “(” and “)” may be replaced by the characters “<” and “>” respectively.

Rules for the file statement are as follows:

The medium for the specified input file(s) and output file is indicated by the keyword DISK, CARD, etc. When the medium is DISK, the absence of a disk-name (mfid) indicates the system disk. CARD refers to 80-column card only. TAPE refers to magnetic tape only. CASSETTE refers to magnetic tape cassette only. DISK refers to any kind of disk-device. The input file for a tagfile or keyfile creation must be on disk.

The PURGE option indicates that the input file(s) are to be purged after use.

The record size and records-per-block values are numeric values. When the input medium is DISK, the record size and records-per-block may be omitted. For a merge specification, input disk file descriptions with record size specifications may be interspersed with descriptions without such specifications.

If the records-per-block is omitted and record size is given, a blocking factor of 1 is assumed.

In all cases (except an index sort), input and output files must have the same record sizes.

The values of record size and records-per-block may be omitted for output files. For a sort, the values assumed are those of the input file. For a merge, the values assumed are those of the first specified input file.

For a keyfile creation sort, the output specification enclosed in parentheses must be the single word KEY-FILE. The output will be on disk and record and block sizes are not user definable.

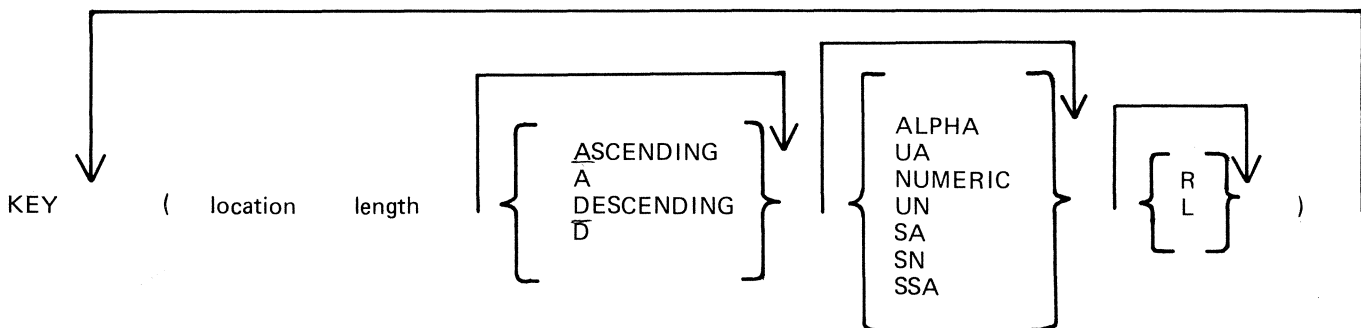
For a tagfile creation sort, the output specification enclosed in parentheses must be the single word TAG-FILE. The output will be disk and record and block sizes are not user definable.

## The Key Statement

This statement defines the record key(s) that are used for the sort or merge.

A number of keys may be specified, each key description being enclosed in parentheses. The first key will be the major key and additional keys will be minor keys of decreasing significance.

Format:



The "location" is a numeric value specifying the position of the key relative to the start of the record, in 4-bit units. The first 4-bit unit has a location of 1. The key location is given by the position of the left-hand 4-bit unit in the key (which, depending on the key format), may be a character or a sign. The key should start on a byte boundary unless a record sort with a numeric key is performed.

The "length" is a numeric value specifying the key length, in 4-bit units. This must include the sign, for signed keys.

The keywords ASCENDING and DESCENDING determine the order of collation. These keywords may be abbreviated to A and D respectively. If omitted, the default is ASCENDING.

The format of the key is specified by one of the following keywords:

- ALPHA or UA – unsigned 8-bit alphanumeric
- NUMERIC or UN – unsigned 4-bit numeric
- SA – signed 8-bit alphanumeric
- SN – signed 4-bit numeric
- SSA – 8-bit alphanumeric with separate sign



The default is ALPHA.

For a signed key, the position of the sign is specified by one of the following keywords

R – right-hand (least significant) end of key

L – left-hand (most significant) end of key

The default in L.

For a description of key types and sign zone interpretation, see later under “KEYS”.

## The User-Option Statement

These statements have three functions:

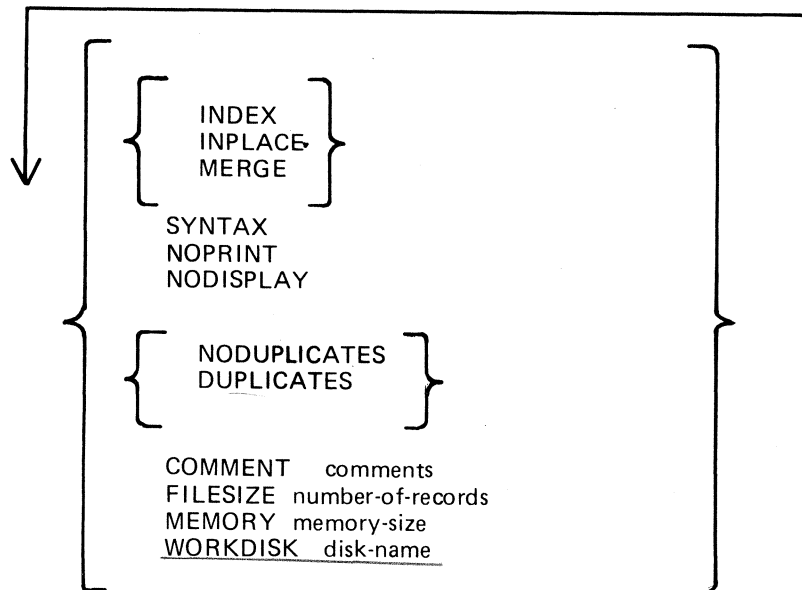
to specify which function is required

to tailor a sort or merge to the particular machine configuration (memory, printer availability, etc).

to add comments

The user-option statements are optional; if more than one are used they may appear in any order relative to each other or to the file and key statements.

Format:



The type of sort is given by one of the keywords INDEX, INPLACE, or MERGE. If one of these does not appear, a regular full record sort is assumed. The keyword INDEX specifies the creation of a keyfile or tagfile, depending on the output file details (see FILE statement). The keyword INPLACE specifies a full record sort using a minimal amount of disk work space. The keyword MERGE specifies a merge of several input files.

Note that if the INPLACE option is specified on a system that has not implemented the in-place sort, then a regular full record sort will be performed.

The keyword SYNTAX specifies that a check on the correctness of the sort statements is to be made without the sort actually being performed.

The keyword NOPRINT stops the listing of the sort statements on the printer. If used, this keyword should be the first entry. If the statements are input via the SPO they are not printed and so this keyword is not required in this case. The NOPRINT option also affects the printing of error and warning messages (see later).

The keyword NODISPLAY controls the display of messages on the SPO during the sort. This option can be used both in initiating messages and from file-oriented statements. It suppresses startup and termination messages. It does not affect the display of error and warning messages. Error and warning free sorts and merges will show no SPO activity if this option is used.

The keyword NODUPLICATES specifies that duplicate keys are not allowed in a keyfile creation. The keyword DUPLICATES specifies that duplicate keys are allowed in a keyfile creation. Both options are valid only when creating a keyfile. If neither is specified, the default is NODUPLICATES.

The keyword COMMENT introduces comment text. The end of the comment text is either the end of the input or the end of a record if the input comes from a starfile. Comments may appear between user-option statements and between file descriptions and key descriptions.

The keyword FILESIZE provides the following capabilities:

specification of sort disk work space where the input file is not on disk (this use is not required if the input is from disk).

specification of maximum size of the output file if on disk.

allowance for future expansion of the output disk file where the sort/merge will not by default create a large enough file.

This keyword should be followed by a number giving the specified maximum number of records. For non-disk output files, the value is used for optimization purposes. If not used, default values are assumed where necessary. This option is not applicable to the inplace sort or to keyfile or tagfile creations.

The keyword MEMORY specifies the amount of non-overlayable work area to be used by the sort. This option is not applicable to the merge or to the inplace sort. If this is not enough for a successful sort, then this option is overridden. The memory size is in bytes; for example, MEMORY 1024.

The keyword WORKDISK enables the regular sort to utilize disk space in an efficient manner. It is not applicable to the merge or inplace sort. When the work-disk is specified, the sort locates up to half the work space on that disk, with the rest on the system disk. If this option is not used, but the input or output file is resident on a user disk, the work space is shared between that disk and the system disk. In all other cases the work space is located entirely on the system disk. The named disk may be any type of disk applicable to the system in use.

## Examples

To sort the system disk file INP.FILE using the key starting at character 5 of length 3 characters, creating a system disk file OUT.FILE:

```
SORT FILE IN INP.FILE (DISK) OUT OUT.FILE (DISK)
      KEY (9 6)
```

To create a keyfile OUTKEY.FILE on disk PR2 from a data file INP.FILE1 on disk PR2, using a 5-byte key starting at the first byte:

```
SORT FILE IN PR2/INP.FILE1 (DISK)
      OUT PR2/OUTKEY.FILE (KEYFILE)
      KEY (1 10)
      INDEX
      COMMENT DUPLICATES NOT ALLOWED
```

To merge the three system disk files FILE1, FILE2 and FILE3 into an output file MERGE.OUT:

**SORT**

**FILE IN FILE1 (DISK) FILE2 (DISK) FILE3 (DISK)**

**OUT MERGE.OUT (DISK)**

**KEY (5 10)**

**MERGE**

## FUNCTIONAL DESCRIPTION

The five functions of the sort are described here:

- Regular record sort.
- Inplace record sort.
- Keyfile creation.
- Tagfile creation.
- File merge.

### Regular Record Sort

All the records contained within the specified input file are ordered using one or more keys. Deleted records (see later) are not included in the output file. See later for details of the keys. Refer to figure 5-1 for an example of a regular record sort, where the key is starting in byte 3 and is 5 characters long, and the sort is in ascending order. The X's refer to any other characters.

The input file must be wholly contained on one hardware type, although it may be a multi-reel or dual-disk file. No other programs may write to this file during the execution of the sort.

The sort uses non-overlayable memory during execution. The amount is calculated according to the input file and key sizes. The amount may be specified as a user option, in which case the specified amount is used unless it is less than enough for a successful sort. In the latter case the specified value will be overridden.

The sort uses disk work space, of up to 2.2 times the size of the specified input file. For the location of the work disk space, refer to the WORKDISK user option (see earlier). This work space is returned to the system at end-of-job.

### Inplace Record Sort

This is the same as the regular record sort, except that the records are sorted within the input file. No new output file is created. The time taken is substantially greater than a regular sort, for the same input specifications. If deleted records are present in the file before the sort, they are removed: hence the number of records in the file may decrease after it has been ordered.

The inplace sort uses non-overlayable memory during execution. The size of this area cannot be specified at initiation.

The input file must be on disk. No other programs may access this file during execution of the inplace sort. The output file must be the same as the file specified for input.

If a particular system does not implement an in-place sort, a regular sort will be performed instead.

The inplace sort uses disk work space, of 0.2 to 0.3 times the size of the input file. When the input file is resident on a user disk, up to one-half of the work space is located on that disk, otherwise all work space is located on the system disk. This work space is returned to the system at end-of-job.

## Keyfile Creation

*use pit value*  
A new file (the "keyfile") will be created containing one record for each record of the input file (the "data file"). The keyfile is sorted in order of the specified keys, and each keyfile record contains the key and a pointer to the corresponding record in the data file. Any deleted records in the data file are not referenced in the keyfile. Note that the records in the data file are not re-ordered and deleted records in the data file are not removed. Refer to figure 5-2 for an example of a keyfile creation, where the key is starting in byte 3 and is 5 characters long, and the sort is in ascending order. The X's refer to any other character.

Duplicate keys are not allowed unless specified (see the user-option statements Duplicates and NODuplicates). If they occur, then the record number is displayed on the SPO for each such occurrence, and the sort will continue but the output keyfile will be purged at end-of-job.

The keyfile creation uses disk work space, of up to 2.2 times the size of temporary file created by the sort in this case. This file is large enough to contain one record with the key value and record number for each record in the input file. For the location of the work disk space, refer to the WORKDISK user option (see earlier). This work space is returned to the system at end-of-job.

*(null)*  
Certain key values are not allowed during a keyfile creation. The key must not consist of all binary zeroes, or must not contain any byte whose value is hex FF. If such a key is encountered, the record number is displayed on the SPO, and the sort will continue but the output keyfile will not refer this record in the data file.

## Tagfile Creation

A tagfile creation is similar to a keyfile creation, except that the output file contains only the record pointers, and not any key values. The tagfile records, however, are ordered in key value order, as specified by the sort. Any deleted records are not referenced in the tagfile. Refer to figure 5-3 for an example of a tagfile creation, corresponding to the keyfile creation in figure 5-2.

A tagfile is a null keyfile. It is suitable for use as an ADDRROUT file in RPG, and for limited indexed access in COBOL (the tagfile is read sequentially).

Disk space requirements are the same as for keyfile creation. *(2.2 times)*

## Merge

The merge merges up to 16 input files, using one or more specified keys, producing one output file. Deleted records in the input files are not included in the output file. If there are duplicate keys values, the order in which they are placed in the output file is given by the order in which the input files are specified.

Each input file must be wholly contained on one hardware type, although it may be a multi-reel or dual-disk file. No other programs may write to these files during the execution of the merge.

Each input file must have the same record size and the same position and length for each key. Each file must be already correctly ordered on the specified keys. If this is not the case, the merge will terminate prematurely after displaying a message on the SPO.

Refer to figure 5-4 for an example of a merge of two files, with a key starting at byte 3 which is 5 characters long. The X's and Y's refer to any character.

The merge uses non-overlayable memory during execution. The size of this area cannot be specified at initiation: it will be approximately equal to the sum of the block sizes of the input files and the output file.

The merge does not use any disk work space.

## Details Of Sort Keys

A “key” is the field within each record that is used for sorting or merging. If several distinct field within a record are specified, then each field is a separate key. The relative order of importance of the keys is determined by the order in which they are specified. Figure 5-5 illustrates this with a two-key sort, using the KEY statement.

KEY (5 6 ALPHA) (15 2 DESCENDING ALPHA)

The X’s indicate any character. In this example the three-byte field is the major key, sorted in ascending order: the one-byte key is a minor key sorted in descending order within the order of the major key.

For a keyfile creation, only one key may be used. This key must be a maximum of 28 bytes long, must be a whole number of bytes in length, and must start on a byte boundary.

For all sorts except keyfile and tagfile creation, there can be up to 10 keys. The sum of the length of all keys (including signs) must be a maximum of 29 bytes.

The available key types are discussed here, under the keyword specified in the KEY statement (see earlier):

### ALPHA (or UA)

Unsigned 8-bit alphanumeric field, containing ordinary ASCII characters. Note that this may consist of the 8-bit ASCII digits “0” to “9” but still be termed alphanumeric. This key type is the default.

### NUMERIC (or UN)

Unsigned 4-bit numeric field, where each 4-bit unit is a binary coded decimal digit, 0000 to 1001 (0 to 9).

### SA

Signed 8-bit alphanumeric field. Each byte is an ordinary ASCII character (including the digits 0-9), except that either the first or the last character indicates the sign. Whether the sign is the first or last character is specified by the keyword L (left) or R (right). The default is L (first character; leading sign). The convention for coding the sign character is given in Table 5-1. These characters are termed “overpunched signs” by analogy with historical punched card systems.

### SN

Signed 4-bit numeric field. Each 4-bit unit is a binary-coded decimal digit, 0000 to 1001 (0 to 9), except that either the first or the last 4-bit unit indicates the sign. Whether the sign is the first or last 4-bit unit is specified by the keyword L (left) or R (right). The default is L (first 4-bit unit); leading sign. The convention for coding the sign is given in Table 5-2.

### SSA

8-bit alphanumeric field with separate sign. Each byte is an ordinary ASCII character (including the digits 0 to 9), with the sign given by an ASCII character in either the first or last character. Whether the sign is given by the first or last character is specified by the keyword L (left) or R (right). The default is L (first character); leading sign. The convention for coding the sign character is given in Table 5-3.

The position of a sign within a signed key (left or right) must be the same throughout all occurrences of the key. Signed keys are ordered so that negative values come before zero and positive values

8-bit keys may start on 4-bit unit boundaries, unless the separate sign type (SSA) is used, or the key is to be used in keyfile or tagfile creation.

## Deleted Records

A deleted record is denoted by every byte in the record (including the key) containing the value hex FF. The action taken by the various sort options is discussed earlier. Deleted records may be physically removed by the FS utility.

## Output Messages

Output messages cover warnings and errors. Messages are generated by both the sort intrinsic and the sort language processor. The intrinsic messages are numbered by event numbers in the same way as MCP output messages (see section ). The sort language processor messages are numbered in a similar way.

Messages can be divided by number as follows:

### 0-99

Sort language processor messages, displayed on the printer. Such messages appearing in the list below that are followed by a series of dots (...) should be read with the phrase NEAR COL XXX (with XXX replaced by an appropriate column number) in place of the dots.

#### 0-34

Warnings, where corrective action is attempted.

#### 35-39

Warnings, where no corrective action is attempted.

#### 40-59

Errors in syntax (that is, the format of the sort statements is incorrect).

#### 60-99

Errors in semantics (that is, an inconsistency has been detected in the statements, such as a key position greater than the record size).

### 170-200

Sort intrinsic messages, displayed on the SPO.

Certain messages may be suppressed by the NOPRINT and NODISPLAY keywords in the sort statements.

The NOPRINT option suppresses listing of the sort statements on the printer by the sort language processor. If this option is set, a maximum of five errors and four warning messages are directed to the SPO, with only the error or warning number being given (no explanatory text). The NOPRINT option has no effect on sort-intrinsic-generated messages.

The NODISPLAY option suppresses display on the SPO of start-up and termination messages by the sort intrinsic. Messages in the list below that are marked with an asterisk (\*) are those that are suppressed when this option is set. Note that it is not possible to suppress individual messages; every applicable message is suppressed if the option is set. The NODISPLAY option has no effect on sort language processor messages.

Number	Message
0	EXPECTED SLASH NOT FOUND, "/" INSERTED ...
1	EXTRA "FILE IN"...
2	MERGE INTRINSIC IGNORES <WORK-DISK OPTION>
3	OVERLENGTH PART OF <LABEL NAME> IGNORED ...
4	INPLACE INTRINSIC IGNORES <WORK-DISK OPTION>
5	EXPECTED BRACKET NOT FOUND, "<" INSERTED ...
6	<DUPLICATE OPTION> VALID IN INDEX-KEYFILE SORT ONLY
7	EXPECTED BRACKET NOT FOUND, ">" INSERTED ...
8	ILLEGAL TO DELETE INPUT FILE, <PURGE OPT> IGNORED
9	OUTPUT BUFFER SIZE TOO BIG, <BLOCK FACTOR> REDUCED ..
10	<USER OPTION> ALREADY INVOKED, LATEST USE ...

Number	Message
11	MERGE <SORT TYPE OPTION> NOT SPECIFIED
12	OVERLENGTH PART OF <DISK NAME> IGNORED ...
13	MISSING "FILE IN" ...
14	INDEX <SORT TYPE OPTION> NOT SPECIFIED
15	EXTRA "KEY" ...
16	<FILE SIZE OPT> VALID FOR MERGE/REGULAR SORT ONLY
17	MISSNG "KEY" ...
18	INPLACE INTRINSIC IGNORES <MEMORY OPTION>
19	<M-FILE/DP ID> IGNORED ON NON-MAGNETIC MEDIA FILE ...
20	NUMBER TOO BIG, MAXIMUM VALUE ALLOWABLE ASSUMED ...
21	not used
22	<SIGN POSITION> GIVEN FOR UNSIGNED KEY ...
23	FIRST UNIT NUMBERED 0 RATHER THAN 1 ...
24	<FILE SIZE OPT> IGNORED SINCE OUT OF RANGE ...
25	MERGE INTRINSIC IGNORES <MEMORY OPTION>
26	<BLOCK FACTOR> OF 0 NOT ALLOWED, 1 ASSUMED ...
27	IN- AND OUT-FILE RECORD SIZES MADE EQUAL
28	<BLOCK FACTOR> TOO LARGE, MAXIMUM ASSUMED ...
29	INPLACE SORT MUST HAVE IDENTICAL IN- AND OUT-FILES
35	IDENTICAL IN/OUT - FILES WILL PRODUCE DUPLICATE FILE
36	NOT NECESSARY TO PURGE CARD FILE ...
37	ALPHANUMERIC KEY DOES NOT START ON BYTE BOUNDARY ...
40	<KEY STATEMENT> ALREADY PROCESSED, NOW ...
41	<DIGIT STRING> EXPECTED ...
42	<CHARACTER STRRNG> EXPECTED ...
43	<SEPARATOR STRING> EXPECTED ...
44	<RCRD-BLCK PAIR> MUST BE GIVEN FOR NON-DISK IN-FILE ...
50	NO <FILE STATEMENT> SPECIFIED
51	ILLEGAL WORD ...
52	<LETTER STRING> EXPECTED ...
53	MISSING <LABLE NAME> ...
54	UNSUPPORTED <IN/OUT MEDIA> ...
55	UNSUPPORTED <SORT TYPE OPTION> ...
56	PART OF <FILE STATEMENT> MISSING, NOW ...
57	NO <KEY STATEMENT> SPECIFIED
58	<FILE STATEMENT> ALREADY PROCESSED, NOW ...
59	FINAL STATEMENT INCOMPLETE ...
60	TOO MANY KEY SPECIFICATIONS ...
61	TOO MANY FILE SPECIFICATIONS ...
62	INPUT FILES RECORD SIZES NOT IDENTICAL ...
63	<RECORD SIZE> OUT OF RANGE ...
64	EXTRA DIGITS IN OVERLENGTH STRING IGNORED ...
65	KEY LENGTH OUT OF RANGE ...
66	MIN LENGTH OF SN KEY IS TWO 4-BIT UNITS ...
67	BUFFER SIZE TOO LARGE ...
68	DUPLICATE <IN-FILE PARAMS>, LATEST INSTANCE ...
69	BUFFER SIZE TOO BIG FOR <IN/OUT MEDIA> ...
70	ONLY ONE IN-FILE LEGAL FROM MULTIPLE TAPE ...
71	MERGE INSTRINSIC NEEDS AT LEAST 2 INPUT FILES
72	INDEX PARAM MUST BE "OUT...<KEYFILE/TAGFILE>"
73	KEY OVER-RUNS RECORD BOUNDARY
74	ILLEGAL TO OVERWRITE INPUT FILE WITH TAG/KEY FILE



Number	Message
75	ALPHANUMERIC KEY LENGTH NOT EVEN NUMBER OF 4-BITS ...
76	<MEDIA> MUST BE DISK FOR IN-PLACE SORT
77	IN- AND OUT-FILE RECORD SIZES MUST BE IDENTICAL
78	INDEX-KEYFILE KEY LENGTH NOT EVEN NUMBER OF 4-BITS
79	ONLY ONE KEY LEGAL IN INDEX-KEYFILE SORT
80	INDEX-KEYFILE SORT KEY TOO LONG
81	INDEX-KEYFILE SORT KEY MUST BE "... A UA/UN>"
82	ONLY INDEX SORT CAN SPECIFY "KEYFILE/TAGFILE"
83	INDEX-KEYFILE SORT KEY MUST START ON BYTE BOUNDARY
84	MIN LENGTH OF SSA KEY IS FOUR 4-BIT UNITS ...
85	SSA KEY MUST START ON BYTE BOUNDARY ...
86	CURRENT SUM OF KEY LENGTHS OUT OF RANGE ...
170	DUPLICATE RECORD <record number>
171	ILLEGAL INDEX KEY IN RECCRD <record number>
172	RECORDS LOST / GAINED BY SORT-MERGE
173	<number> DUPLICATE RECORDS
174	<number> RECORDS CONTAINING INVALID INDEX KEYS
175*	<number> DELETED RECORDS
176*	<number> RECORDS MERGED
177*	<number> FILES MERGED
178	SORT-MERGE OUTPUT FILE NOT CREATED
179	SORT-MERGE ABNORMAL EOJ
180	SORT-MERGE SOFTWARE ERROR
181*	<number> RECORDS REFERENCED BY KEYFILE/TAGFILE
182	NO INITIATING MESSAGE
183*	<number> RECORDS SORTED
184	FILE ERROR <<number>> NEAR RECORD <record number> ON <file name>
185	UNORDERED MERGE INPUT FILE <file name> NEAR RECORD <record number>
186	TOO MANY RECORDS FOR SORT-MERGE
187	DUPLICATE RECORDS-KEYFILE NOT BUILT
188	INITIATING MESSAGE INVALID
189*	SORT-MERGE VER x.y.z INITIATED FROM <mix number>/ <program name>
193	INPUT RECORD SIZES UNEQUAL - BAD FILE <filename>
194	IN/OUT RECORD SIZES BAD - OUTPUT SIZE CHANGED
195	BAD RECORD/BLOCK SIZE FOR OUTPUT DEVICE
196	KEY OVER-RUNS RECORD END
197	CANNOT SPLIT INDEX FILE
198	<number> PARITY BLOCKS
199	INDEX INPUT FILE NOT TYPE DATA

Message 184 represents differing file errors depending upon the value of <number>. Defined meanings are as follows:

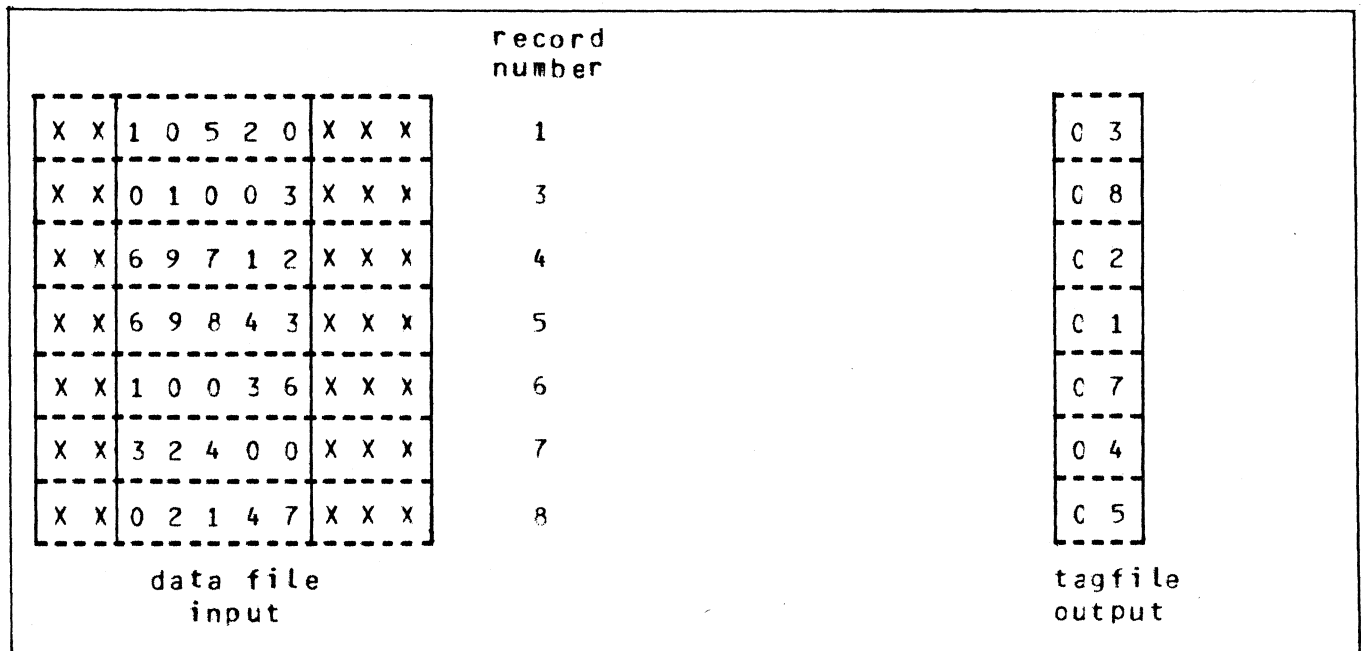
- |                           |  |
|---------------------------|--|
| 1 - EOF on output file    | 7 - output file error                    |
| 2 - parity on input file  | 8 - parity on sort workfile              |
| 3 - EOF on sort workfile  | 9 - parity on input file (block ignored) |
| 4 - parity on output file |  |
| 5 - sort workfile error   |  |
| 6 - input file error      |  |

			record number			
X X	1 0 5 2 0	X X X	1	X X	0 1 0 0 3	X X X
X X	1 0 0 3 5	X X X	2	X X	0 2 1 4 7	X X X
X X	0 1 0 0 3	X X X	3	X X	1 0 0 3 5	X X X
X X	6 9 7 1 2	X X X	4	X X	1 0 0 3 6	X X X
X X	6 9 8 4 3	X X X	5	X X	1 0 5 2 0	X X X
X X	1 0 0 3 6	X X X	6	X X	3 2 4 0 0	X X X
X X	3 2 4 0 0	X X X	7	X X	6 9 7 1 2	X X X
X X	0 2 1 4 7	X X X	8	X X	6 9 8 4 3	X X X
data file input				data file output		

Figure 5-1. Regular Record Sort

			record number			
X X	1 0 5 2 0	X X X	1	0 3	C 1 0 0 3	
X X	1 0 0 3 5	X X X	2	0 8	0 2 1 4 7	
X X	0 1 0 0 3	X X X	3	0 2	1 0 0 3 5	
X X	6 9 7 1 2	X X X	4	0 6	1 0 0 3 6	
X X	6 9 8 4 3	X X X	5	0 1	1 0 5 2 0	
X X	1 0 0 3 6	X X X	6	0 7	3 2 4 0 0	
X X	3 2 4 0 0	X X X	7	0 4	6 9 7 1 2	
X X	0 2 1 4 7	X X X	8	0 5	6 9 8 4 3	
data file input				keyfile output		

Figure 5-2. Keyfile Creation



**Figure 5-3. Tagfile Creation**

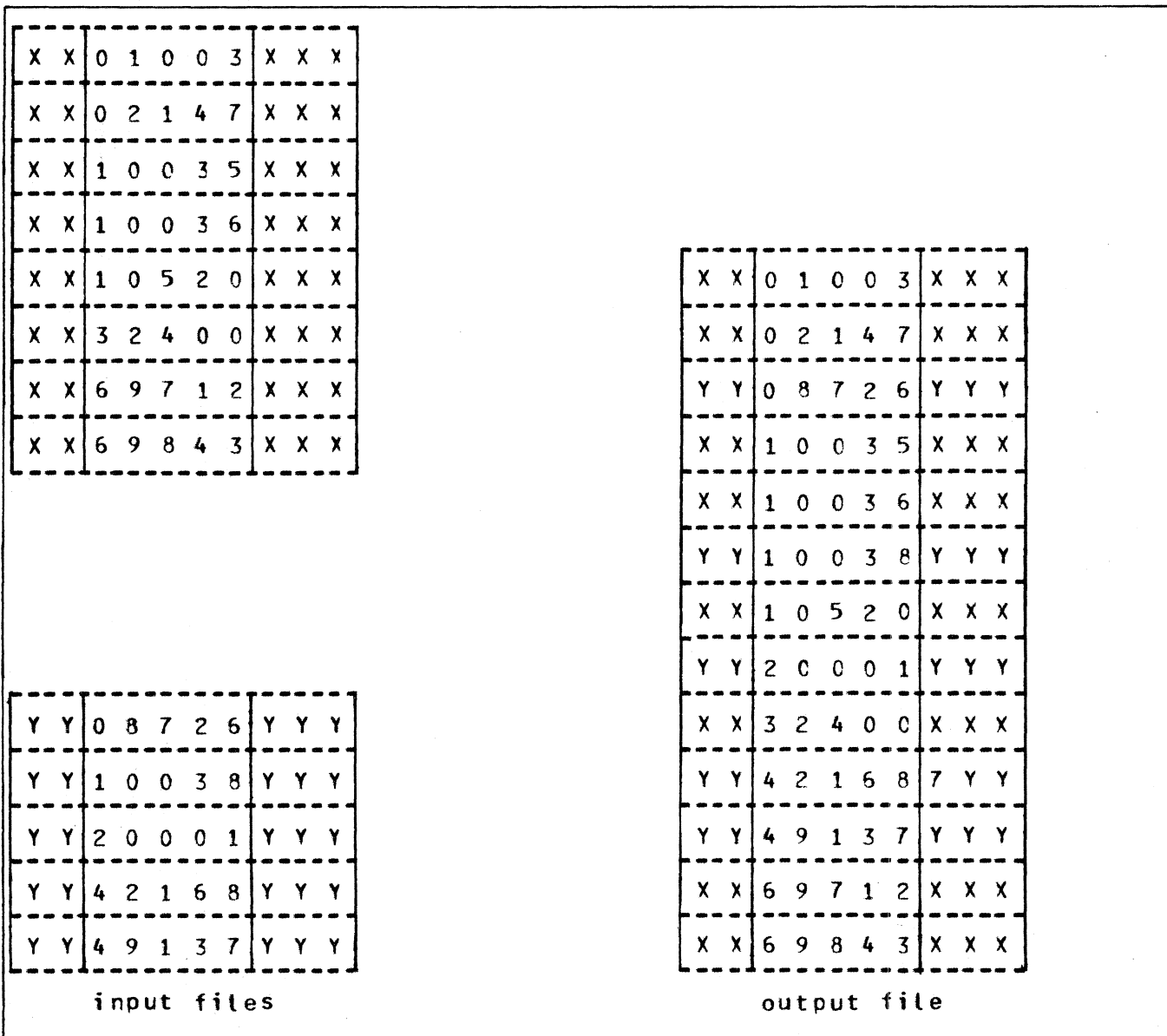
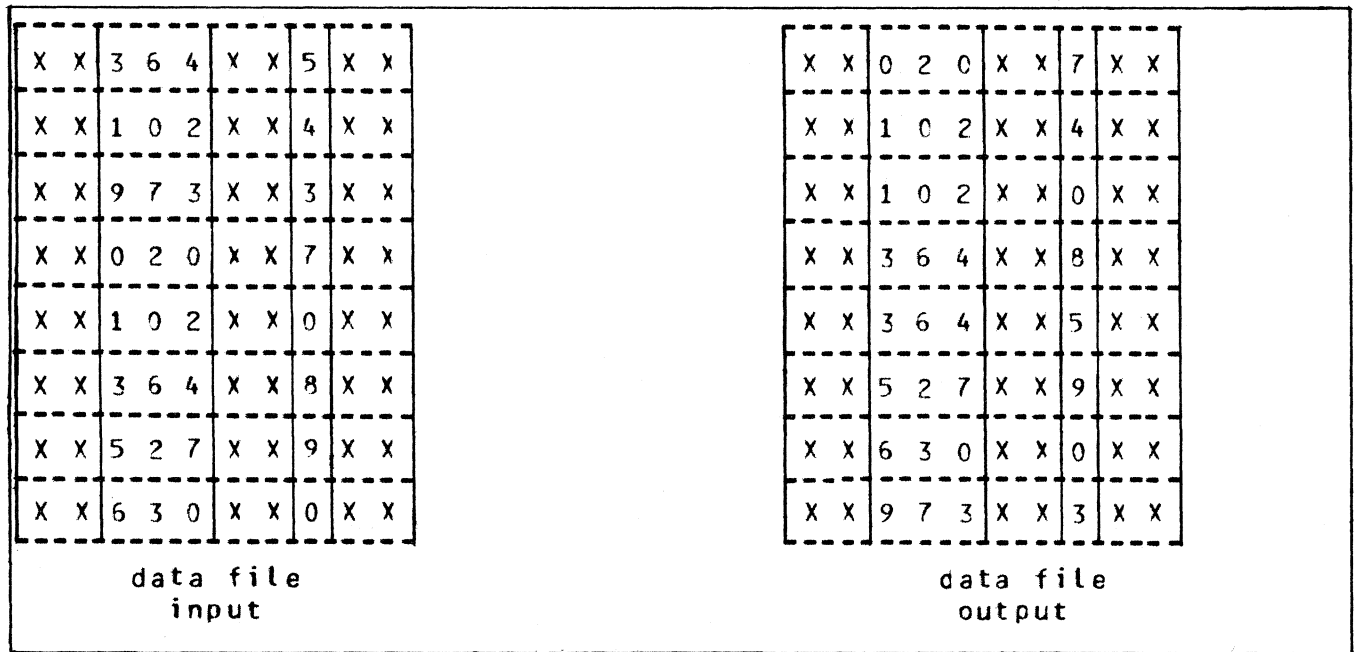


Figure 5-4. File Merge



**Figure 5-5. Multiple Key Sort**

**Table 5-1. Sign Convention For Signed 8-Bit Alphanumeric Fields**

Key Value	Hex Code	ASCII Character
-0	2D	-
-0	7D	-
+0	30	0
+0	7B	0
-1	4A	J
-2	4B	K
-3	4C	L
-4	4D	M
-5	4E	N
-6	4F	O
-7	50	P
-8	51	Q
-9	52	R
+1	31	1
+2	32	2
+3	33	3
+4	34	4
+5	35	5
+6	36	6
+7	37	7
+8	38	8
+9	39	9

Note: any other hex code in the sign character is interpreted as positive, with the key value given by the binary value of the right-hand 4 bits of the character.

**Table 5-2. Sign Convention For Signed 4-Bit Numeric Fields**

Key value	Binary Code (BCD character)
negative	0101 (5)
positive	0011 (3)

Note: any value other than 0101 (5) is interpreted as positive.

**Table 5-3. Sign Convention for Separate Sign Character with 8-bit Alphanumeric Fields**

Key value	ASCII character (hex value)
negative	"-" (2D)
positive	"+" (2B)

Note: any character other than "-" is interpreted as positive.

# SECTION 6

## COMPILATION FACILITIES

### INTRODUCTION

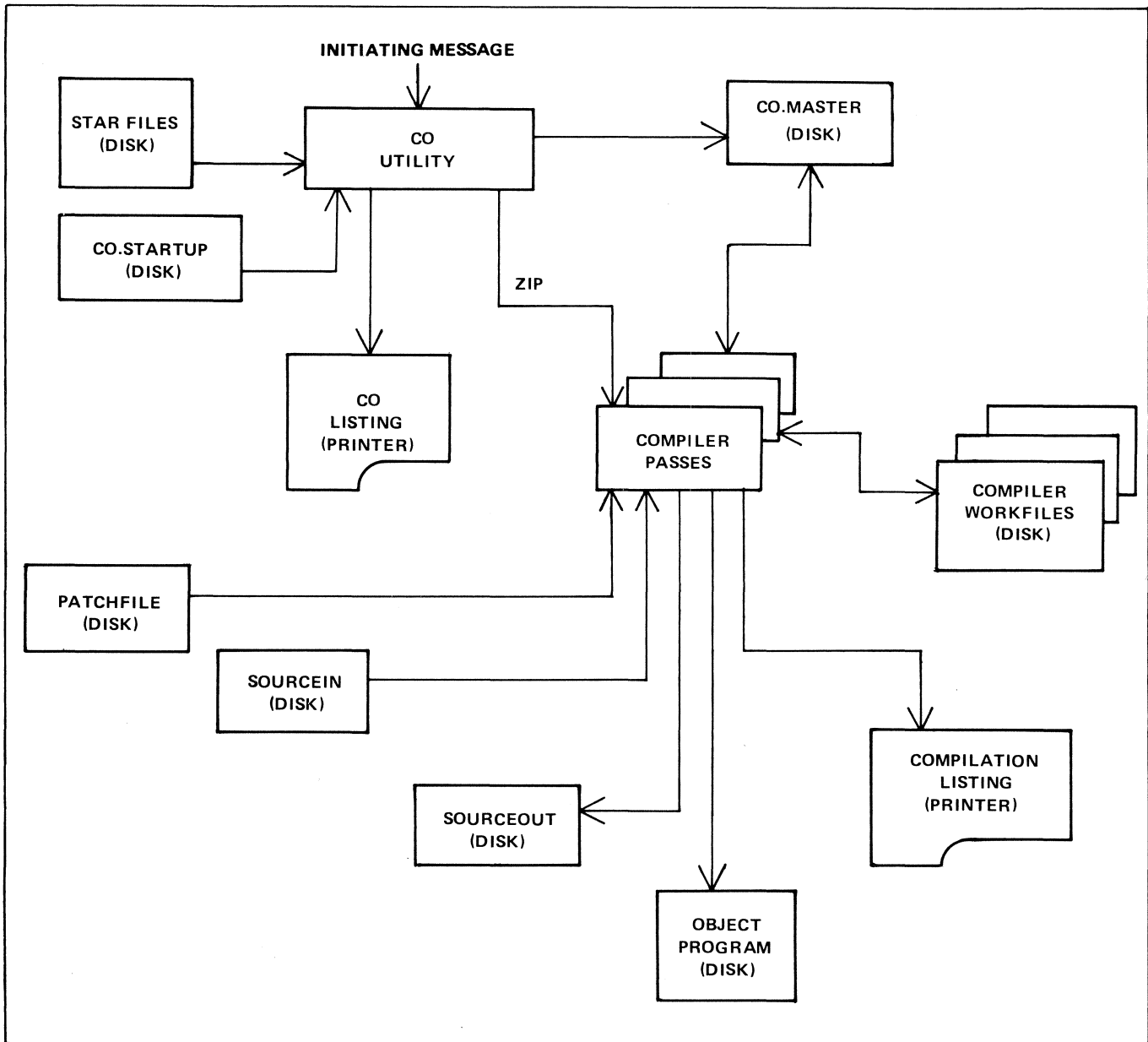
Compilation of programs written in CMS COBOL, RPG and MPL can be performed with the CO (compile) utility. CO is a normal utility program residing on disk. It is used to co-ordinate the various parts of the CMS COBOL, RPG and MPL compilers. Each compiler consists of several object code files (called "passes") and produces a number of workfiles to pass information between each pass. The CO utility allows initial input to be made to the compiler by specifying such things as input and output file names.

Additionally, CO allows multiple executions of each compiler by storing compiler workfile information in a master file called CO.MASTER on the system disk. The compiler passes have access to the information in this disk file. Information in this file also allows the CO utility to perform restarts if the system halts during a compilation. This restart facility eliminates the need to rerun a compilation from pass one if one or more passes have already completed successfully.

CO uses some standard names for input and output files, which can be changed by the initial CO message. The basic CO operation is given in Figure 6-1.

Initial input to CO is either from the initiating SPO message or through macro (star) files or through a disk file called 'CO.STARTUP' on the system disk. CO generates the CO.MASTER file used to co-ordinate the compiler passes. There is an option to produce a CO listing. Information provided to CO enables the user to describe the following:

- input patch file
- input source file
- output source file
- output object program
- output compilation listing
- compiler workfiles



**Figure 6-1. Operation Of CO Utility**

CO provides the ability to set “dollar options” for the compilation.

CO operates in two basic ways

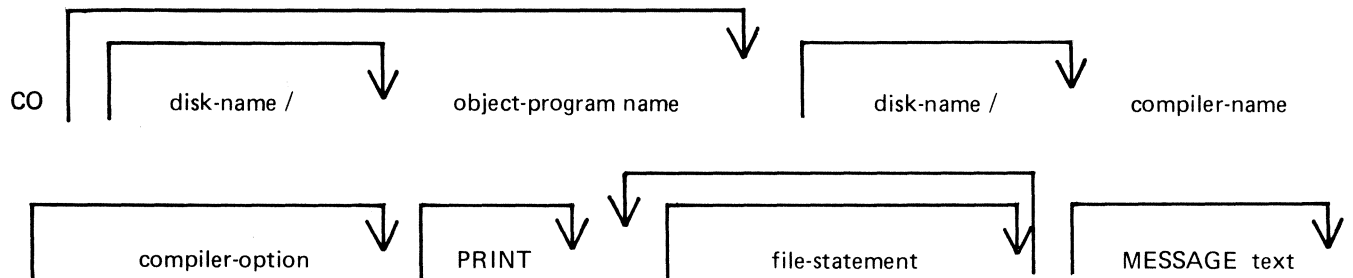
to initiate and control a single compilation.

to interrogate compilation status, and restart or clear an aborted compilation.



## TO INITIATE A SINGLE COMPILATION

Format:



This initial input can be entered on the SPO when executing CO. CO can also be started with no additional information from the SPO: in this case the information is either on a card file or on a disk file called "CO.STA-RTUP" on the system disk. Selected parts of the input can be provided as star files see later in this section. All disk files used for inputting information to CO can be 80 byte card-image records created by CMS CANDE or CREATE, with all information in the first 8 records and the first 72 characters of each record.

Semi-colons may separate any clauses after the compiler-option, for readability.

The object-program-name is optional. If not specified, it will be created on the system disk with a name as follows:

- "COBJECT" – for COBOL compilations
- "RPGOBJECT" – for RPG compilations
- "MPLOBJECT" – for MPL compilations

The name of the disk for the generated object program may also be specified. If not specified, the system disk is assumed.

The compiler-name may be one of the following:

- COBOL – the COBOL compiler
- RPG – the RPG compiler
- MPL – the MPL compiler
- RPGXREF – the separate RPG cross-reference program
- OPTLIST – the separate COBOL/RPG optimizer

If the optional disk-name is given before the compiler-name, then the compiler may be executed from a user disk so long as all passes are on the specified disk. If no disk name is given then the compiler must be on the system.

The compiler-option may be one of the following:

- SYNTAX (abbreviation SY)
- LIBRARY (abbreviation LI)
- GO (alternative SAVE)

With a "compile for syntax", no object program is generated. For COBOL and RPG, COBSVERTER is not executed; for MPL, the compilation stops at the end of pass 3.

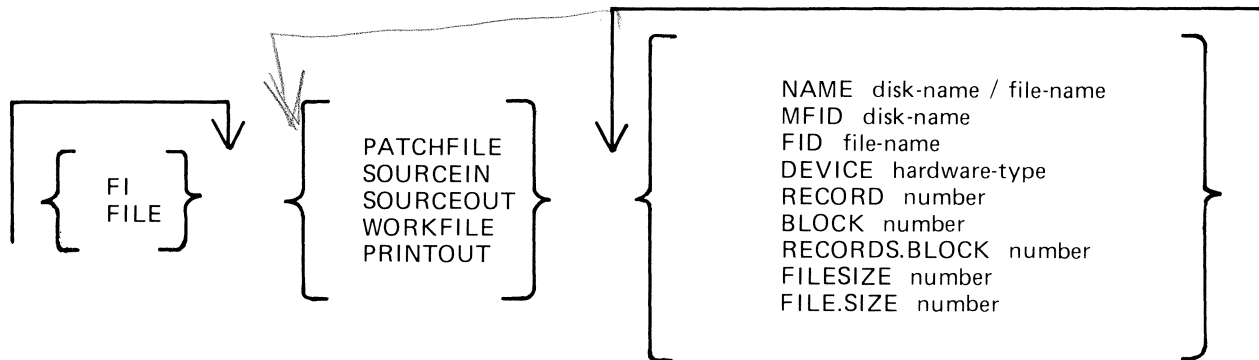
With a “compile to library“, an executable object program is created if there are no syntax errors, and saved on disk with the object-name as specified in the CO statement. A compile to library is the default compiler-option.

A “compile and go” is the same as a compile to library, with the additional feature that the object program is executed. The CO utility goes to EOJ after zipping the object program.

If the PRINT option is specified, an edited listing of the startup message is output, along with a list of settings for all the file parameters which are modifiable through CO. A list of default dollar card settings for the compilation is output along with any dollar options entered via the CO message. This object listing also contains a log of all error messages displayed.

The file statements allow names and other attributes to be set for the compiler input and output files. The general form of these statements are given here, but not all attributes may be set for all files these will be discussed in turn.

Format:



The meaning of the file attributes are as follows:

#### NAME

This specifies the CMS file name plus disk name and may be of the form MFID/FID.

#### MFID

This specifies the disk name of the given file. The default is the system disk. The MFID must be a maximum of 7 characters.

#### FID

This specifies the file name of the given file. The default is the system disk. The FID must be a maximum of 12 characters.

(Note: FID and MFID may be used together, or NAME may be used instead of the MFID/FID combination).

#### DEVICE

This specifies the type of peripheral of the input or output file. The default for the PRINTOUT file is line printer. The default for all other files is any disk. The device is specified by a 2-character mnemonic as follows:

```

card readers:
    any card reader                AR
    any multi-function unit        AM
    80-column reader                R8
    80-column reader/punch         M8
    96-column reader                R9
    96-column reader/punch         M9
card punches:
    any card punch                  CP
    any multi-function unit        AM
    80-column punch                 P8
    80-column reader/punch         M8
    96-column punch                 P9
    96-column reader/punch         M9
tapes and cassettes (NRZ or PE)
    any tape                         AT
    write-disabled reel             MT
    write-disabled cassette         CT
    (alternatives: CS, CASSETTE)
Note: if a write-disabled device is specified
for an output file, CO will substitute a
write-enabled device in the specification, and
issue warning...
printers:
    any printer                     AP
    serial printer (console)        SP(A)
    line printer                     LP(A)
disks:
    any disk, default cartridge     DK
    (alternatives: DC, DISK)
    Burroughs Super Mini            DM
    Fixed disk                       DF
    pack                             DP

```

## FILESIZE

This attribute specifies the maximum number of records in the output file. If used with PATCHFILE, the number specifies the total number of source lines to be compiled, including dollar cards and "COPY statements" in the case of COBOL compilation. The number may be followed by the letter K to denote thousands. For example, the statement

```
FI SOURCEOUT FILESIZE 4 K
```

specifies a filesize of 4000 records. A space must separate the "K" from the number.

An alternative spelling of the keyword is "FILE.SIZE".

The files are as follows:

## PATCHFILE

This is the primary source input file, and contains dollar records and source records which may optionally be merged with a secondary input file to produce an output source file. Attributes which may be set, and their default values are:

```

NAME                PATCHFILE
MFID                 system disk (0000000)
FID                  PATCHFILE
DEVICE               DK
FILESIZE             0

```

## SOURCEIN

This is the optional secondary input file. Attributes which may be set, and their default values, are:

NAME	SOURCEIN
MFID	system disk (0000000)
FID	SOURCEIN
DEVICE	DK

## SOURCEOUT

This is the optional source output file produced by merging PATCHFILE and SOURCEIN. Attributes which may be set, and their default values, are:

NAME	SOURCEOUT
MFID	system disk (0000000)
FID	SOURCEOUT
DEVICE	DK
FILESIZE	4 K
RECORD	80
RECORDS.BLOCK	9
BLOCK	720 for disks 240 for cassettes 2000 for magnetic tapes

## WORKFILE

This refers to the intermediate workfiles produced by the compiler during the compilation. The only attribute that can be specified, and its default, is:

MFID	system disk (0000000)
------	-----------------------

## PRINTOUT

This refers to the listings produced by the various compiler passes during the compilation. The only attribute that can be specified, and its default, is:

DEVICE	LP
--------	----

## The MESSAGE statement

The reserved word MESSAGE indicates the start of the list of dollar options for that compile. The text consists of a list of one or more dollar cards, taken from the list given below, separated by spaces.

For example, for an MPL compilation the following would be valid:

```
MESSAGE $LIST $XMAP $SEGMENT FR20
```

Use of this feature is not valid for COBOL compilations.

## Examples:

To compile the COBOL source program AR678S, and create the object program AR678, both on the system disk:

```
CO AR678 COBOL LI FI PATCHFILE FID AR678S
```

To merge the RPG patch file RQ20P with the source file RQ20S, and create a new source RQ20SN and object RQ20NEW, all on disk RDEV:

```
CO RDEV RQ20NEW RPG LI; FI PATCHFILE NAME RDEV/RQ20P;  
FI SOURCEIN NAME RDEV/RQ20S; FI SOURCEOUT NAME  
RDEV/RQ20SN; MESSAGE $MERGE
```

To compile the MPL source program MTEST.S from the disk USR1 to produce object MTEST on disk USR1, with CO listing, and compiler listing on the console:

```
CO USR1 MTEST MPL LI; FI PATCHFILE NAME USR1/MTEST.S;  
PRINT; FI PRINTOUT DEVICE SPA; MESSAGE $LIST
```

To compile source PATCHFILE with COBOL and create object COBOBJECT on the system disk:

```
CO COBOL
```

To patch COBOL source CS500 (found on tape) with cardfile CRDPATCH and produce an object program CNEW on disk F1, putting compiler workfiles on disk FSCRATCH:

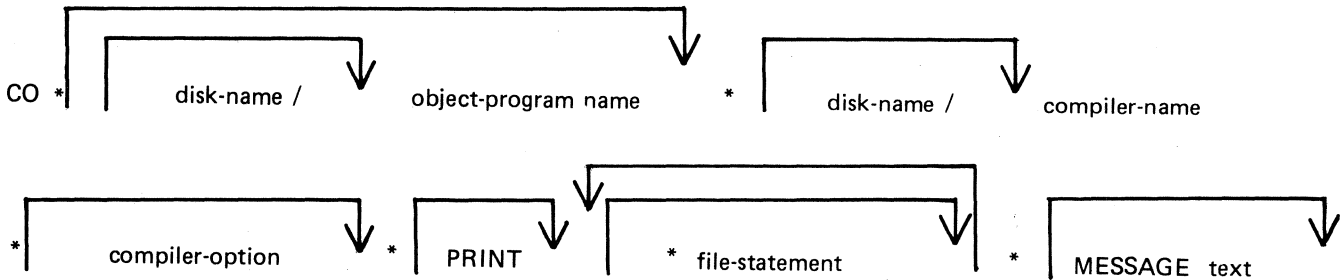
```
CO FI/CNEW COBOL LI;  
FI PATCHFILE FID CRDPATCH DEVICE AR;  
FI SOURCEIN FID CS500 DEVICE AT;  
FI WORKFILE MFID FSCRATCH;
```

## USE OF MACRO CALLS

All or part of the initiating message to CO may be provided as data in disk files. This is indicated by an asterisk (star) in the message.

Following the asterisk must be a valid file name, including the disk-name if not on the system disk. When the initiating message is scanned, the contents of a star-file are included in the scan. At the end of the star-file contents, scanning continues with the primary message. The complete initiating message, or individual clauses, may be included in the star-files. The message format for CO has been repeated below with an asterisk where a star-file may be used:

Format:



Example:

Consider four star-files, with names and contents as follows:

name	contents
---	-----
FILE1	AA RPG GD
FILE2	PATCHFILE NAME BB
FILE3	DEVICE AR
FILE4	SOURCEOUT FID CC

Then the following two initiating messages to CO are valid:

```
CO *FILE1 PRINT *FILE2 DEVICE AR
CO *FILE1 PRINT *FILE2 *FILE4
```

but the following two messages are invalid, because the contents of FILE3 is wrong, as it starts in the middle of a file-statement:

```
CO *FILE1 PRINT *FILE2 *FILE3
CO *FILE1 PRINT PATCHFILE NAME BB *FILE3
```

## COMPILER DOLLAR OPTIONS

In the following list, (D) indicates the options which are set by default. This notation is also used in the output listing from CO, to distinguish default settings from deliberate dollar card settings.

For COBOL, no dollar options may be set or reset by CO. For a fuller description of the available RPG and MPL options, consult the relevant compiler manual.

### RPG

\$ LIST	(D)	\$ NLIST
\$ SEQ		\$ NSEQ
\$ XMAP		\$ NXMAP
\$ SUPR		\$ NSUPR
\$ LOGIC		\$ NLOGIC
\$ MERGE		\$ NMERGE
\$ SEVERE	(D)	\$ NSEVERE
\$ MAP		\$ NMAP
\$ NAMES		\$ NNAMES
\$ NEW		\$ NNEW

Any RPG dollar options specified in the startup message for CO will override any occurrences of that option appearing in either the PATCHFILE or SOURCEIN input files.

### MPL

\$ LIST	\$ CODE
\$ NOLIST	\$ NEWTAPE
\$ LISTE	\$ FORMAT
\$ LISTP	\$ SEGMENT file-name
\$ SEQUENCE	\$ SEGSIZE integer
\$ XMAP	
\$ NOWARNING	

Any MPL dollar options specified through CO are used only as initial settings, and may be overridden by dollar cards appearing in either of the source input files.

## TO INTERROGATE THE STATUS OF COMPILATIONS

CO may be used to determine the status of any compilations. This is done by interrogating the CO.MASTER file. The message is:

CO MX

This function may be used at any time, provided that the mix is not already full. It yields information in the following form:

CO	MIX-NUMBER	n
	COMPILER NAME	compiler-name
	CURRENT COMPILER PASS	compiler pass name
	PROGRAM NAME	object-file-name
	WORKFILE CHARACTER	"A"

(where "A" is a character used by CO in the meaning of workfiles so that different compilations can precede with separate workfiles). In addition to the above information, if a compilation is either awaiting Restart or Clear (see later), or has been Restarted but has not yet gone to end-of-job, the following will be displayed:

(THIS CO IS IN RESTART MODE)


Note that the "CO MX" function will give information about all current compilations marked in the CO.MASTER file, even if they were prematurely stopped by a system failure. This is different from the MCP "MX" intrinsic, which will only give the jobs currently in the multiprogramming mix.



## TO RESTART AN ABORTED COMPILATION

This function may be used to restart a compilation which has been terminated prematurely due to a system failure such as a clear start or ZIP failure. The message is:

Format:

CO RESTART 

If only one compilation was being done, then the simple message

CO RESTART

is sufficient. The compilation is resumed at the beginning of the pass in which the failure occurred. If more than one compilation was being done, the mix-number must be supplied, which is the mix-number of the particular CO that was running at the time of failure. This number is determined either from the SPO log, or the CO MX facility.

If the failure occurred during the printing phase of a COBOL compilation (pass COBOL7), then this pass can be restarted at a specified line-number. The line number must be in the range 000000 to 065535. For example, the message

CO RESTART 2 010000

will cause the CO with mix-number 2 to be restarted, which will cause COBOL7 to start printing at line 010000.

Once a RESTART has been initiated, no new compilations can be started until the restarted compilation has gone to normal EOJ. If that is not possible, for instance because a file is not present, then the restarted job should be CLEARed (see later) to allow other compilations to be initiated.

Other CO versions executing are undisturbed by a RESTART operation.

When a restarted job terminates, whether naturally or as a result of a CLEAR (see later), the message "RESTART COMPLETED"

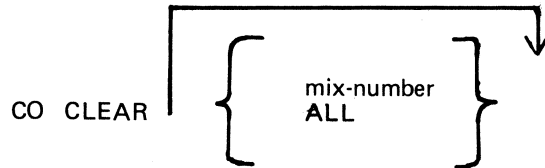
is displayed on the SPO. If a new CO, having the same mix-number as the one that failed, is started up before the failed CO can be restarted or cleared, then the block of information in CO.MASTER for the failed CO is lost and that compilation cannot be restarted. In this case the following message is displayed:

"CO MIX-NUMBER n CANNOT NOW BE RESTARTED  
COMPILATION BLOCK RE-USED".

## TO CLEAR AN ABORTED COMPILATION

If a clear start or other failure has occurred, and it is decided not to Restart compilations, then on or all of the compilations may be cleared. References to one or all of the compilations are deleted from the CO.MASTER file, and workfiles belonging to the compilation are removed unless the compiler requested that they be saved. The message is:

Format:



Providing a mix-number clears only the compilation whose controlling CO had the specified mix-number. The keyword ALL clears all compilations known to CO. If only one compilation was being done, then the simple message

```
CO CLEAR
```

is sufficient.

Other CO versions executing are undisturbed by a CLEAR operation.

Any CO can be CLEARed at any time whether or not a restart is in operation or pending.

Example of aborted compilation:

Assume that a system failure occurred while doing a COBOL compilation. The controlling CO had a mix-number of 6 and the compiler was in the OPTLIST pass. Assume also that, since the failure, an RPG compilation had been initiated.

The message

```
CO MX
```

may result in the following information:

```
CO MIX NUMBER          4  
  COMPILER NAME        RPG  
  CURRENT COMPILER PASS  RPGPHASE1  
  WORKFILE CHARACTER    D  
  
CO MIX NUMBER          6  
  COMPILER NAME        COBOL  
  CURRENT COMPILER PASS  OPTLIST  
  WORKFFLE CHARACTER    C  
      (THIS CO IS IN RESTART MODE)
```

Then to restart the COBOL compilation, enter

```
CO RESTART 6
```

to cause the compilation to start at the beginning of OPTLIST. If the COBOL compilation is not required to be restarted, then enter

```
CO CLEAR 6
```

Note that in both cases the RPG compilation is unaffected.

## ZIP FAILURES

If a zip failure occurs, or a particular compiler pass is DS'ed or DP'ed, CO displays one of the messages in table 6-1 indicating the reason for the failure, then takes one of the following two actions:

If no Restart is in operation or pending.

One of the following messages is displayed:

'CO SHOULD BE RESTARTED OR CLEARED'.

'CO MIX-NUMBER n MUST BE RESTARTED OR CLEARED'.

The CO utility is forced into "Restart mode" which prevents any new COs or any other RESTARTS being performed until the CO in question has been either restarted and completed, or cleared.

If a Restart is in operation or pending.

One of the following messages is displayed:

"CO SHOULD BE RESTARTED OR CLEARED"

"CO MIX-NUMBER SHOULD BE RESTARTED WHEN  
EXISTING RESTART COMPLETE".

The CO utility is not forced into "Restart mode" in this case. It is required to RESTART the job as soon as possible after the existing restart is complete.

**Table 6-1, Zip Failure Messages**

ZIP FAILURE DUE TO PROGRAM FILE NOT FOUND FOR program-name
ZIP FAILURE DUE TO INTERPRETER FILE NOT FOUND FOR program-name
ZIP FAILURE DUE TO NO MEMORY FOR program-name
ZIP FAILURE DUE TO NO USER DISK FOR program-name
ZIP FAILURE DUE TO MIX FULL FOR program-name
ZIP FAILURE DUE TO USER COUNT ERROR program-name
ZIP FAILURE DUE TO DUPLICATE PACK program-name
ZIP FAILURE DUE TO INVALID LOAD REQUEST program-name
ZIP FAILURE DUE TO MCS ALREADY PRESENT program-name
ZIP FAILURE DUE TO DISK ERROR program-name
ZIP FAILURE DUE TO CODE FILE ERROR program-name
ZIP FAILURE DUE TO ILLEGAL DATA REQUEST program-name
ZIP FAILURE DUE TO REASON UNKNOWN program-name
COMPILER PASS DS'ed program-name
COMPILER PASS DP'ed program-name

## RESERVED WORDS

All keywords used in initiating messages to CO are reserved words; that is, they cannot be used for file names or other user-defined parts of the initiating message. A complete list of the reserved words is given in table 6-2.

Table 6-2. CO Reserved Words

MPL	SOURCEOUT
COBOL	WORKFILE
RPG	PRINTOUT
RPGXREF	NAME
OPTLIST	MFID
SAVE	FID
GO	DEVICE
SYNTAX	FILE.SIZE
SY	FILESIZE
LIBRARY	RECORD
LI	BLOCK
PRINT	RECORDS.BLOCK
MESSAGE	MX
FILE	RESTART
FI	CLEAR
PATCHFILE	ALL
SOURCEIN	

## ERROR MESSAGES

The printout file for CO may contain error messages from the CO utility itself. These are listed in table 6-3, and are largely self-explanatory.

Table 6-3. Error Messages from CO

ERROR NO	ERROR MESSAGE
1	First word in Initiating Message is not a valid disk or file name.
2	Invalid object program name or compiler name specified.
3	Warning - Compiler Pack Id. truncated to 7 characters.
4	No compiler specified.
5	Warning - Program Pack Id. truncated to 7 characters.
6	Warning - Program Id. truncated to 12 characters.
7	Reserved Word not found when expected.
8	FI/FILE not followed by one of PATCHFILE, SOURCEIN, SOURCEOUT, WORKFILE, PRINTOUT.
9	Reserved Word used out of context.
10	Warning - Compiler Option already modified - previous mod. ignored.
11	Warning - Print Option specified more than once.
12	Record size * Blocking factor > 65535, Block size set equal to Record size.
13	MESSAGE not used by COBOL - Message Text ignored.
14	Warning - Message Text already processed - this Message Text ignored.
15	This dollar option may not be Set/Reset via the Compile Utility.
16	\$SEGMENT file-name parameter not a valid file name.
17	CO.MASTER no longer on disk - job terminated.
18	Warning - \$SEGMENT pack-id truncated to 7 characters.
19	Warning - \$SEGMENT program-name truncated to 12 characters.
20	\$SEGSIZE parameter not numeric.
21	Modification not valid for SOURCEIN - modification ignored.
22	Warning - MESSAGE contains no valid dollar cards.
23	Warning - \$card not valid for this compiler.
24	Warning - Invalid \$card in Message Text.
25	Warning - PATCHFILE modifications already done - these mods ignored.
26	Warning - PATCHFILE modifications incomplete.
27	Modification not valid for PATCHFILE - modification ignored.
28	Warning - No PATCHFILE modifications.
29	MFID/FID modification already done for this file - mod ignored.
30	Pack name/File name in NAME specification invalid.
31	Pack name in NAME specification truncated to 7 characters.

32 File name in NAME specification truncated to 12 characters.  
 33 Pack name in MFID clause invalid.  
 34 Pack name in MFID clause truncated to 7 characters.  
 35 File name in FID clause invalid.  
 36 File name in FID clause truncated to 12 characters.  
 37 RECORD size of zero specified - mod ignored - default used.  
 38 DEVICE modification already done for this file - this modification ignored.  
 39 Specified Device Type invalid - default device type used.  
 40 FILESIZE already modified for this file - this mod ignored.  
 41 FILESIZE parameter not numeric.  
 42 BLOCK size already modified - this mod. ignored.  
 43 RECORDS.BLOCK/BLOCK parameter not numeric.  
 44 RECORD parameter not numeric.  
 45 SOURCEIN modification already done - these mods ignored.  
 46 Warning - SOURCEIN modification incomplete.  
 47 No SOURCEIN modifications.  
 48 SOURCEOUT modifications already done - these mods ignored.  
 49 Warning - SOURCEOUT modifications incomplete.  
 50 Warning - No SOURCEOUT modifications.  
 51 Specified RESTART mix-number not numeric.  
 52 Specified mix-number out of range.  
 53 Cannot initiate another RESTART - previous RESTART incomplete.  
 54 No CO with specified mix-number available for RESTART.  
 55 Null Initiating Message - no file CO.STARTUP - job terminated.  
 56 Nested Macro Call found.  
 57 Invalid file name in Macro Call.  
 58 Pack name in Macro Call truncated to 7 characters.  
 59 File name in Macro Call truncated to 12 characters.  
 60 Record size for card device > 96, RECORD and BLOCK size set to 96.  
 61 File CO.STARTUP empty - No Initiating Message - job terminated.  
 62 Cannot run CO on this system, SYSTEM STATUS Communicate not supported.  
 63 CO.MASTER not found - job terminated.  
 64 Compiler not zipped - errors in Initiating Message.  
 65 Specified CLEAR mix-number not numeric.  
 66 Specified CLEAR mix-number out of range.  
 67 No CO's to RESTART/CLEAR - job terminated.  
 68 No CO with specified mix-number available for Clearing.  
 69 SYSTEM STATUS Communicate failure - job terminated.  
 70 WORKFILE modification already done - this mod ignored.  
 71 WORKFILE not followed by MFID - mod ignored.  
 72 Specified RESTART line-number for listing not numeric.  
 73 Specified Restart Line-number > 65535 - Line-number set to 0.  
 74 RESTART mix-number not specified.  
 75 No parameter specified for CLEAR function.  
 76 Line-number parameter only valid for COBOL - ignored.  
 77 Illegal character encountered in Initiating Message.  
 78 Cannot continue compilation - Compilation Block

- initialized.
- 79 Non-input Device specified for PATCHFILE - mod. ignored.
  - 80 Non-input Device specified for SOURCEIN - mod. ignored.
  - 81 Macro file not found - <file-name>
  - 82 Non-output Device specified for SOURCEOUT - mod. ignored.
  - 83 BLOCK size for SOURCEOUT is not a multiple of the RECORD size.
  - 84 Specified BLOCK size for Tape Device too large - default value used.
  - 85 Specified RECORD size for Tape Device too large - default used.
  - 86 80 col card device RECORD size > 80, RECORD and BLOCK size set to 80.
  - 87 96 col card device RECORD size > 96, RECORD and BLOCK size set to 96.
  - 88 Warning - PRINTOUT modification already done - this mod. ignored.
  - 89 Device specified for PRINTOUT not Printer Type - ignored.
  - 90 PRINTOUT not followed by DEVICE - mod. ignored.
  - 91 BLOCK size or Blocking-factor of zero specified - default used.
  - 92 SOURCEOUT FILESIZE specified as zero - mod. ignored - default used.
  - 93 Specified PATCHFILE FILESIZE > 65535 - default value used.
  - 94 Specified SOURCEOUT filesize > 65535 - default value used.
  - 95 Specified RECORD size > 65535 - default value used.
  - 96 Specified BLOCK size or Blocking-Factor > 65535 - default used.
  - 97 When compiling MPL on Cumbernauld Support System, level number required.
  - 98 If 80 Col. card output device used, record truncation will occur.
  - 99 RECORD size already modified - this mod. ignored.
  - 100 Macro file present but empty - <file-name>

## RESTARTING EXECUTING CO VERSIONS

Note that CO does not prevent restarting or clearing a currently-executing compilation. However, when the original version of the compiler pass, that was executing when the RESTART was done, terminates and returns to CO, CO will recognize that the block of information in the CO.MASTER file has been overwritten. The message

“CANNOT CONTINUE COMPILATION-COMPILATION  
BLOCK INITIALIZED”

is displayed and the job is terminated. This may result in problems such as the occurrence of duplicate files. If an executing compilation is CLEARed, there is an added complication that any workfiles for that compilation are removed.

Restarting or clearing any currently-executing compilation is therefore not advised.



# SECTION 7

## NUMBERED SYSTEM SOFTWARE OUTPUT MESSAGES

### **INTRODUCTION**

The CMS software maintains a table of output messages used by the MCP, the interpreters, and other items like the SORT intrinsic. Each of these messages is given a number called the "event number". The event number is always displayed, enclosed in brackets, as part of the message.

This section lists each message in event number order.

## EVENTS # 1-9

### Software Information

These are information indicating error conditions. If any action is required, other message(s) will follow immediately.

#### Message format:

mix number/program-name <EVENT#> file-name peripheral function message ERROR WHILE IN verb status.

The peripheral is given by the mnemonic for example, DK for disk. The function is INPUT or OUTPUT. The message is given in the table below. The verb is OPEN, CLOSE, READ, WRITE, etc. The status gives additional information such as the disk address for disk failures.

#### Examples:

```
12/PD <46> MYDISK/SYSMEM DM OUTPUT TIMEOUT ERROR  
... WHILE IN OPEN 0033
```

```
10/MYPROG <2> FR203D DM READ PARITY ERROR  
... WHILE IN READ 14A5
```

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
1	TAPE FORMAT	Ending label on labelled tape is missing or invalid.	None. Program has indicated it can handle the error itself.
2	PARITY <STATUS>	Hard parity error found on this device.	None. Program has indicated it can handle the error itself. <STATUS> shows disk sector address for disk errors.
3	TIMEOUT <STATUS>	For tape device, no data was found for quite a distance on tape (exact length depends upon unit); For disk, cylinder specified could not be used.	None. Program has indicated it can handle the error itself. <STATUS> shows disk cylinder address for disk errors.
4	ADDRESS <STATUS>	Sector could not be found on disk (cylinder was found). Either software has indicated an invalid sector or sector address on disk is corrupt.	None. Program has indicated it can handle the error itself. <STATUS> shows disk sector address for disk errors.

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
5	REWIND	Problem encountered on tape rewind.	Retry the program.
6	DESCRIPTOR	Identifies I/O error that is not PARITY, TIMEOUT, ADDRESS or REWIND.	None. Program has indicated it can handle the error itself.
7	NON-FILE FULL ON MFID	It is possible for all entries in available table (non-file directory) to be in use. If a file then releases its disk space it may not be possible to enter it in the non-file directory. Disk is checkerboarded.	Run SO utility and then retry the program.
8	DEVICE LOCKED	A non disk device is closed with lock.	None.
9	DEVICE NOT PURGED	An attempt is made to close purge a write disable magnetic tape drive.	None.

## EVENTS # 10-19

### Software Suspensions

When a program that is running encounters a condition that prevents it from continuing, MCP will suspend the program and inform the operator as to the reason for the suspension. When the condition is cleared, MCP will normally allow program to continue running. If program does not continue automatically then the operator should issue a "GO" command (see "GO" intrinsic) to the program.

#### Message formats:

mix number/program-name <EVENT #> WAITING file-name device message

#### Examples:

10/LIST <17> WAITING UNLAB LISTPRT AP NO FILE

10/COPY <17> WAITING TL REEL 001 AT NO FILE

02/NGD03 <12> WAITING FILE 255 NO USER DISK

05/MPL.SEG <10> WAITING CARDFL DF NO FILE

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
10	NO FILE	Disk or tape file this program needs to use cannot be found.	Check for correct disk or tape; Supply program with backup copy of file requested;  Program will continue when file is supplied.
11	DUPLICATE FILE	While attempting to place a certain file on disk, program has discovered a file with the same name already exists on disk. Program halts, as 2 files with same name cannot reside on one disk.	Normally remove with RM the existing file from disk. Program will continue. If in doubt refer to program instructions.
12	NO USER DISK	There is no more available space on disk; or space available is insufficient to hold file this program is attempting to write; or disk is "checked".	With KA, analyze amount of available space remaining. If disk is filled remove with RM any unnecessary files and program will continue; or

(cont'd)

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
	NO USER DISK (cont'd)		If disk is filled and a dual-pack file is desired, assign different disk to this program (see AD intrinsic); or if disk is checkerboarded use SQ utility to consolidate disk space, then re-execute program that encountered suspension.
13	DIRECTORY FULL	When the disk was initialized the disk directory was constructed to contain a fixed number of file names. The directory has now reached its capacity.	Remove with RM any unnecessary files and program will continue; or DS the suspended program. Replace disk with another disk having sufficient directory space, and re-execute the program.
14	DEVICE NOT AVAILABLE	Output device that program needs in order to continue processing is either unavailable or not ready.	RY required device; or assign program to alternate device (see AD intrinsic).
15	FORMS REQUIRED	Program is waiting for operator to insert correct forms into the output device before it will continue processing.	Insert correct forms into output device. Then use "AD" intrinsic to assign device to program (see AD intrinsic).
16	GENERATION NUMBER MISMATCH	While opening an old indexed file, a discrepancy has been found between the generation number fields of the keyfile and the data file. The operating system will cancel the suspension when the discrepancy no longer exists.	

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
17	NO FILE	Program needs a device that is either unavailable, busy, or not ready.	RY required device; or allow program currently using the device to go to End of Job and suspended program will then automatically be able to use device;
18	DUPLICATE FILE	2 or more disks (or tapes) having the same names have been found on-line. Only one disk (or tape) of a given name may be on-line at a time.	Check for correct disk or tape. Replace (or relabel) one of the duplicates.
19	FILE IN USE	The specified file cannot be opened because it has already been opened by the maximum number of users allowed for the desired move i.e. input/output.	Wait until the file is not in use.

## EVENTS # 20-40

### INVALID REQUEST ON CLASS A OR B COMMUNICATE TO MCP

These messages normally indicate program errors. Program in error should be DS'ed or DP'ed (see DS and DP intrinsics), if necessary. Then operator should attempt to run the program again. If the same error is encountered, request technical assistance.

Message format:

mix number/program-name <EVENT#> CANNOT verb text parameter

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
20	(NULL)	Illegal verb.	
21	NOT FIB	Byte 1 of Priority A or B communicate should contain Data Segment Table Index of a File Information Block. Data Segment Table does not contain File Information Block.	
22	FILE LOST	The medium which contained the file has been illegally physically removed when it was still required (e.g. half close).	None
23	ATTRIBUTE MISMATCH	Each class A communicate is <u>checked against the files attributes</u> . Attribute includes such things as <u>device, myuse, access mode etc.</u> This message indicates failure on this check e.g. <u>read on an output file, start on a non-disk file, write on a closed file etc.</u>	
24	BAJ SEQUENCE	Priority A error may be noted when OPENED I/O with Sequential Access 1. REWRITE was not immediately preceded by successful READ.	

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
		<p>2. OVERWRITE was immediately preceded by OPEN or START communicate.</p> <p>3. OVERWRITE or REWRITE was preceded by conditional READ which failed.</p>	
25	BAJ WORK AREA	<p>Work Area Segment specified within File Information Block cannot be used, because it either indicates FIB segment or it is Read Only segment - yet communicate requests data transfer to that segment.</p>	
26	ILLEGAL KEY	<p>Key requested on Priority A communicate for a disk was equal to zero.</p> <p><b>**NON-RESIDENT COMMUNICATE HANDLER**</b></p>	
27	DEVICE NOT ON SYSTEM	<p>There is no device of the required kind on the system.</p>	
29	DUAL PACK MISMATCH	<p>The two parts of a dual pack file have been found to be inconsistent.</p>	
30	ALREADY OPEN	<p>Communicate requested an OPEN on already opened file.</p>	
31	ALREADY CLOSED	<p>Communicate requested a CLOSE of already closed file.</p>	
32	BAJ ADVERB	<p>Adverb to OPEN was determined to be illegal for any use of the following:</p> <p>1. MYUSE equal to "0"</p>	



EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
		2. MYUSE incompatible with device (for example, I/O for LP). 3. Access mode Random for non-disk device. 4. Access mode not equal to Sequential or Random.	
33	BAD BLOCK OR RECORD SIZE	Record and/or Block Size been determined to be incompatible or illegal for any of following reasons: 1. Buffer or Record length equal to zero for new disk or tape files. 2. Record length exceeds physical Block size. 3. Buffer length not an integer multiple of Record length. 4. FPB or DFH buffer length not multiple of 180 if old file is opened with specified Buffer length other than zero. 5. DFH Buffer length not multiple of FPB Buffer length if old file is opened with specified Buffer length.	

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
34	BAD FILE SIZE	Maximum file size exceeds 1048560 or 65535 if single area File.	
35	BAD NUMBER OF BUFFERS	Number of buffers specified exceeds 16.	
36	BAD DEVICE	Device requested not supported by CMS. Device code is illegal.	
37	BAD FILE TYPE	1. An OPEN has been requested on "SYSTEM" file by unprivileged user program. 2. Unprivileged user program requested a CLOSE with Lock or Purge of file with SYSTEM VMFILE or Firmware File Type.	
38	PROTECTION ERROR	Privileged user attempted to CLOSE with Lock or Purge a VMFILE or Firmware file while the file was still in use.	
39	BAD FILE-NAME	OPEN or CLOSE has detected an illegal special character imbedded in file-name.	
40	BAD KEYSIZE	During OPEN of a new index file, MCP has discovered in File Parameter Block (FPB) a key length of zero or one that exceeds 28 bytes.	

## EVENTS # 41-49

### Fatal Device Errors

These messages indicate fatal hardware errors. The program encountering the error should be DS'ed or DP'ed (see DS and DP intrinsics) if necessary. Then the operator should attempt to run program again. If the same error is encountered the media (disks, tapes, etc) involved should not be used further until field engineering has been notified and media has been checked.

#### Message format:

mix number/program-name <EVENT#> file-name peripheral function message ERROR WHILE IN verb status

The format of these messages is the same as the format for event numbers 1-9.

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
41	BAD LEVEL NUMBER	The implementation number in a keyfile KFFB is greater than this implementation.	
42	DISK LOCKED	Disk write error occurred at file close time. This could be either while writing last block, or updating Disk File Header or available table.	
43	END OF TAPE	Unexpected end of tape encountered.	
44	TAPE FORMAT	Ending label or labelled tape is missing or invalid.	
45	PARITY <STATUS>	Hard priority error discovered on device. <STATUS> shows disk sector address for disk errors.	
46	TIME OUT <STATUS>	For tape device, no data was found for quite a distance on tape (exact length depends upon unit); For disk, cylinder specified could not be found. <STATUS> shows disk cylinder address for disk errors.	

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
47	ADDRESS <STATUS>	Sector could not be found on disk (cylinder was found). Either Software has indicated invalid sector or sector address or disk is corrupt. <STATUS> shows disk sector address for disk errors.	
48	REWIND	Problems encountered on rewinding a tape.	
49	DESCRIPTOR	Identifies I/O error that is not PARITY, ADDRESS, or REWIND.	

## EVENTS # 50-69

### Load Failures

These messages indicate that the MCP failed to begin the processing of a particular program for some reason. Operator should correct the condition that caused the load failure and then try running the program again.

Message format:

[EVENT#] LOAD FAILURE message

Examples:

[53] LOAD FAILURE NO USER DISK

[51] LOAD FAILURE PROGRAM NOT FOUND

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
50	DISK NOT FOUND	Specified disk is not on-line or ready.	Check for correct disk; RY disk if not ready
51	PROGRAM NOT FOUND	Specified file was not found on disk.	Check input and re-input if necessary; Check for correct disk.
52	FULL MIX	System cannot accept any additional programs of this Priority Class at the moment; or program presently running will not permit any other programs to enter the mix.	Allow one program to come to End of Job (EOJ). Then begin required program again; or DS unwanted program(s), if applicable. Then begin required program again.
53	NO USER DISK	There is not enough room on disk for this program's Virtual Memory file. (VMFILE).	Remove with RM any unnecessary files;
54	INTERPRETER NOT FOUND	For B 90:  COBOLINT or BILINTERP were not found on system disk or have become corrupted.	Supply, with COPY, the missing interpreters from backup medium; Replace, with COPY, the corrupted interpreters from backup medium;

(cont'd)

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
	INTERPRETER NOT FOUND (cont'd)	For B 800:  MCP may be corrupted; program operator is attempting to run may be corrupted.	Replace, with COPY, corrupted MCP from backup medium; Replace, with COPY, corrupted program with backup medium;
55	USER COUNT ERROR	Too many programs are trying to use a file. Limit is seven.	Wait until one or more programs goes to EOJ. Then try again.
56	CODE FILE ERROR	Data file name was mistaken for program name, and an attempt was made to run the file instead of a program.	Check input and re-enter.
57	INVALID LOAD REQUEST	Typing error. (EX: too many characters in program name).	Check input and re-enter.
58	INSUFFICIENT MEMORY	There is not enough room in memory (to hold this program's Task Control Block and Program Control Block).	Request technical assistance.
59	MCS ALREADY PRESENT	Only one MCS may be in the mix.	None.
60	DUPLICATE PACK	2 or more disks with the same name are currently on-line, and this system does not permit this.	Re-name one of the disks with RL; check for correct disk.
61	NULL MIX REQUIRED	Specified program may be run only if no other programs are in the mix e.g. SQ	Wait until all programs go to EOJ before running this program.

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
62	ILLEGAL DATA COMM LOAD REQUEST	An NDL task can be loaded only as part of the data comm loading sequence. Any other attempt to load an NDL task is failed.	
63	DISK ERROR	There has been an irrecoverable disk error while attempting to load a program.	
64	UTILITY BUSY	The operator has tried to initiate a super-utility function while another is in the mix.	Wait until the first function ends.
65	INSUFFICIENT REAL STORE	The program run structure cannot be constructed in the amount of memory specified in the real store field of the EX command.	
66	DUAL ALPHABET NOT SUPPORTED	The program requires interpreter and MCP facilities to support dual alphabet or reverse escapement and the current system does not.	

## EVENTS # 70-99

### System Errors

These messages indicate system errors. The program encountering the error should be DS'ed or DP'ed (see DS and DP intrinsics) if necessary. Then the operator should attempt to run the program again. If the same error is encountered, technical assistance should be requested.

Message format:

mix number/program-name <EVENT#> SLICE# message

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
70	SEGMENT OUT OF RANGE	Segment number exceeds number of segments declared in the program.	
71	SEGMENT SIZE ERROR	Offset and length value exceeds declared size of segment.	
72	STACK OVERFLOW	Amount of Control Stack requested during a run has exceeded declared Stack Size.	Use MODIFY utility to increase control stack size of program, and rerun program.
73	STACK UNDERFLOW	Designated code has attempted to retrieve more information from Control Stack than is present.	
74	INSUFFICIENT REAL STORE	The program has attempted to exceed the memory size specified at load time in the real-store field.	
75	PROGRAM TOO LARGE	The program has attempted to exceed the physical memory size.	
80	BAD KEY POSITION	The offset of key position is more than record size.	



# EVENTS # 100-169

## Program Errors

These messages indicate program errors. The program in error should be DP'ed (see DP intrinsics) if necessary. The source program, object program, dumpfile produced by DP, and any data files used by this program should be saved. The operator should try to run the program again. If the same error is encountered, request technical assistance, and supply all relevant data saved to the technician.

Message format:

mix number/program-name <EVENT> SOURCE REFERENCING program segment # segment address program counter address message

Examples:

03/MYPROG <121> 0 SEGMENT 0 ADDRESS  
... 15 SUBSTRING ERROR

04/PROGB <140> 2639 SEGMENT 18 ADDRESS  
... 436 CODE FILE ERROR

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
*ALL INTERPRETERS* :			
100	COMMUNICATE ERROR	MCP returned @80@ in Byte 0 of Fetch Mess- age on a communicate.	
101	COMMUNICATE EOF ERROR	MCP returned an End of File indication in Fetch message (@20 20 00@) and the user has not specif- ied any action if EOF occurs.	
102	COMMUNICATE I/O ERROR	MCP returned I/O error indication in Fetch message (@20 30 00@) and the user has not specified any action to take if error occurs.	
103	SEGMENT NUMBER ERROR	Interpreter detected an invalid code on data segment number.	
104	WRITE ERROR	Interpreter has detect- ed attempt to WRITE into a Read-Only Segment or literal. Code file has become corrupt or an error exists in compiler or interpreter.	

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
105	SEGMENT BOUNDARY VIOLATION	Interpreter, in calculating an address, has discovered that the address of the data or code is out of range. Code file has become corrupt or an error exists in compiler or interpreter.	
**INTERPRETER for MPLII**			
110	INVALID OP	Code file has become corrupt or error exists in MPLII compiler or interpreter.	
115	DESCRIPTOR ACCESS CODE	Program tried to store the fetch value to a non-character field, or to convert to a non-character field, or to store to a self-relative descriptor, or an error in the SETNAME procedure, or a decimal add error.	
116	SEGMENT SIZE ERROR	Segment length is too long or is set while segment is in core.	
117	ADDRESS ERROR	SETNAME extent error; identifier has become out of scope	
118	MESSAGE REFERENCE ERROR	Message reference field not divisible by 4, or illegal access to message reference space.	
119	STRING STORAGE ERROR	Illegal destination in store string instruction.	
120	REMAP ERROR	Program tried to remap a bit descriptor.	

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
121	SUBSTRING ERROR	Attempt made to substring into non-character area.	
122	INDEX ERROR	Program tried to index a self-relative descriptor, or to bit index a self-relative descriptor.	
123	EXIT ERROR	Local data returned from a function.	
124	CPA ERROR	Error in communicate parameter area: for example, message referenced expected, or size of character field at least 3 bytes expected.	
125	DIVIDE ERROR	Divide by zero attempted.	
126	ZIP ERROR	Error returned after ZIP (not used on B80)	
127	BIT DESCRIPTOR ERROR	Bit field overlaps more than one byte boundary.	
128	FPB ERROR	Error in file parameter block encountered - subfield of a non-F.P.B. segment requested.	
129	CONTROL STACK ERROR	Control Stack overflow or underflow.	
130	DATA STACK ERROR	Illegal descriptor encountered.	
131	DECLARATION MODE ERROR	Size of character field greater than 255.	
132	DATA STRUCTURE ERROR	Insufficient room for structure nesting or size of character field greater than 255, or insufficient room	

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
	<b>DATA STRUCTURE ERROR (cont'd)</b>	for array bound evaluation, or non-character descriptor encountered when character descriptor expected.	
<b>**INTERPRETER for COBOL and RPG**</b> -----			
140	INVALID S-OP CODE	Invalid S-Op-Code. Code file is corrupt or error exists in COBOL/RPG compiler or interpreter.	Same as event 143
141	INVALID COPX - GREATER THAN SIZE OF COP TABLE	Invalid COPX - greater than size of COP table. Code file corrupt or error in COBOL/RPG compiler or interpreter.	Same as event 143
142	ALPHANUMERIC FIELD TYPE NOT 8-BIT UNSIGNED	Alphanumeric field type not 8-bit unsigned. Code file corrupt or error in COBOL/RPG compiler or interpreter.	Same as event 143
143	INVALID EDIT MICRO OPERATOR	Invalid EDIT micro Operator. Code file corrupt or error in COBOL/RPG compiler or interpreter.	Re-run program from backup copy. If still in error, request technical assistance.
144	INLINE EDIT MASK NOT CORRECTLY TERMINATED	Inline EDIT MASK not correctly terminated. Code file corrupt or error in COBOL/RPG compiler or interpreter.	Same as event 143
145	EXAMINE SOURCE FIELD ERROR	EXAMINE source field error. Code file corrupt or error in COBOL/RPG compiler or interpreter.	Same as event 143
146	EXAMINE PARAMETER FIELD NOT 8-BIT UNSIGNED, ONE CHARACTER	EXAMINE parameter field not 8-bit unsigned character. Code file corrupt or error in COBOL/RPG compiler or interpreter.	Same as event 143

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
147	EXAMINE CONTROL BYTE ERROR	EXAMINE control byte error. Code file corrupt or error in COBOL/RPG compiler or interpreter.	Same as event 143
148	COMPARE FOR CLASS-CLASS AND FIELD TYPE INCOMPATIBLE	COMPARE for CLASS-CLASS and FIELD type incompatible. Code file corrupt or error in COBOL/RPG compiler or interpreter.	Same as event 143
149	SUBSCRIBED OR INDEXED SUBSCRIPT OR INDEX	Code file corrupt or error in COBOL/RPG compiler or interpreter.	
150	INDEXED/ SUBSCRIBED VARIABLE IS INDEXED/ SUBSCRIBED BY MORE THAN 3 VARIABLES	The array has more than three subscripts or indexes.	
151	FETCH COMMUNICATE RESPONSE FIELD NOT OF LENGTH 3 BYTES		
152	INVALID EXAMINE SPECIFICATION		
160	PERFORM STACK OVERFLOW	Indicates too many PERFORMS without a return for the Perform Stack specified at compile time (if not specified then the default value was used). If this did not result from a programming error, the Perform Stack should be increased.	The size of the perform stack may be increased by the MODIFY utility, using the CONTROL.STACK option to change the PPB (see section 4).
161	NON-POSITIVE OVERFLOW	Subscripts must be positive.	

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
162	ARRAY BOUND VIOLATION	Subscripts outside upper bound of OCCURS clause.	
164	TRANSLATION SOURCE ERROR		
166	INVALID SIGN CODE		
167	I/O ERROR	Invalid read/write to a file.	
168	SORT OR MERGE ERROR	An error has been encountered in SORT or MERGE.	
169	ZIP FAILURE	Zip to another task has failed.	

## EVENTS # 170-199

### SORT Exception Events

Any of these output messages may be displayed while SORT and SORTINTRINS are running. For reporting SORT errors: all SORT errors should be accompanied by either the Sort Spec or the COBOL program that requested SORTINTRINS, as well as the input file(s). Since SORTINTRINS is a machine-dependent program, the method of getting a dump may vary. To get a program dump on the B 80: rerun the program with "GT" on. This will cause SORTINTRINS to dump its run structure on the console printer.

#### Message format:

mix number/program-name <EVENT#> message

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
170	DUPLICATE RECORD record number	Only for Keyfile creation. Another record in file has same key as this record.	If duplicates are desired, specify "DUPLICATES" in input.
171	ILLEGAL INDEX KEY IN RECORD record number	Only for keyfile creation. Either the Key field is all binary 0 or has one or more bytes with hex FF. This record will not be referenced from the keyfile.	None.
172	RECORDS LOST OR GAINED BY SORT-MERGE	Probably indicates an error in SORTINTRINS.	See introductory paragraph concerning errors.
173	number DUPLICATE RECORDS	Normal message tells the operator the total number of records that have duplicates. (See Event 170).	None.
174	number RECORDS CONTAINING INVALID INDEX KEYS	Normal message tells operator total number of records with in- valid index keys. (See Event 171 for further information).	None.
175	number RECORDS DELETED	Records with hex FF in every byte position will be deleted by SORT. Informs operator how many records were deleted.	None.

EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
176	number RECORDS MERGED	Normal message for MERGE only. Tells operator total number of records merged from all files.	None.
177	number FILES MERGED	Normal message for MERGE only. Tells total number of files merged. Should be same as number of files requested in the Sort Specs.	None.
178	SORT-MERGE OUTPUT FILE NOT CREATED	SORTINTRINS was DS'ed; may indicate corrupt SORT or SORTINTRINS programs.	See introductory paragraph concerning SORT errors.
179	SORT-MERGE ABNORMAL EOJ	Early termination due to errors.	
180	SORT-MERGE SOFTWARE ERROR	Error in SORTINTRINS	See introductory paragraph concerning SORT errors.
181	number RECORDS REFERENCED BY KEYFILE- TAGFILE	Only for keyfile-tagfile creation. Tells number of entries of keyfile/tagfile.	None.
182	NO INITIATING MESSAGE	SORT INTRINSIC requires a properly coded Initiating Message. This should be properly formatted by SORT or Sorts within programming languages such as COBOL. Probably indicates an attempt to execute SORT INTRINS directly.	None.
183	number RECORDS SORTED	Normal message tells number of records sorted by the SORT utility. SORT successful.	None.



EVENT	MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
184	FILE ERROR <number> ON SORT-MERGE FILE file- name	The <number> means: 1. EOF on output File 2. PARITY on Input File 3. EOF on Sort Work- file 4. Bad Disk Address 5. SORT workfile Error 6. Input File Error 7. Output File Error Except for 2, this probably indicates error in SORTINTRINS.	See introductory paragraph concern- ing SORT errors.
185	UNORDERED MERGE INPUT MERGE FILE file-name	Files to be merged were found not to be increasing/decreasing key value. Either the file is incorrect or the key position has been incorrectly spe- cified.	SORT files for order. Then retry.
186	TOO MANY REC- ORDS FOR SORT- MERGE	Machine-dependent limitation.	See introductory paragraph concerning SORT errors.
187	DUPLICATE RECORDS - KEYFILE NOT BUILT	Was not specified and duplicates exist. The keyfile will not be built and this message will be displayed.	Specify DUPLICATES in input.
188	INIT MESSAGE INVALID	Initiating message to the SORTINTRINS is not in proper format. This could possibly be caused by a fault in the program that zipped the SORTINTRINS.	See introductory paragraph concerning SORT errors.
189	SORT-MERGE INITIATED FROM mix number/prog- ram-name	Informs the operator which program used the SORT intrinsic.	None.

# SECTION 8

## B 90 DEPENDENT SYSTEM SOFTWARE

### INTRODUCTION

This section covers those items in the CMS software which are operationally different on the B 90 series from other CMS implementations. These differences are mainly a result of the different hardware features involved. The software covered includes:

- Powering the B 90 on and off.
- The B 90 CMS disk "bootstrap" feature.
- Stand-alone utilities, operating without MCP control.
- Loading the MCP.
- Particular features of the B 90 MCP.
- Taking memory (system) dumps
- Utilities released only for B 90 systems.
- B 90 system errors and symptoms.

### POWER ON

Ensure no disks or cassettes are in the system (failure to do this may result in subsequent media corruption).

Turn the system power on. It is then under the control of ROM which performs a mini-test of critical machine components to verify it is capable of starting. The successful completion of this test is verified by the PK lights lighting and then turning off sequentially, with PK1 and PK2 remaining lit.

PK1 permits the loading of a cassette into the system. Some examples of cassettes to load would be (1) AE 500 firmware to cause the B 90 to perform as an AE 500, or (2) ACSYS SL5 emulator firmware cassettes to cause the B 90 to process Series L cassette programs on disk. B 90 cassette loads are not used in CMS.

PK2 permits the loading of information from a disk into the system. Some examples of disks to load would be (1) ACSYS (SL5 emulator) firmware disks, or (2) CMS disks to cause the B 90 to load CMS firmware.

Load the CMS disk in any available disk drive.

A mini-disk is considered loaded immediately the drive unit door is closed, and the blue indicator is lit (disk properly inserted and up to speed). For MCP control the disk must be write-enabled (red indicator lit).

For cartridge disk, wait about 20 seconds for the cartridge to come up to speed (you hear a click as the heads access the disk). Ideally they should be run initially for a few minutes before use to achieve correct running temperature. For MCP control the disk must be write-enabled.

For fixed disk, wait until the 'READY' half of the "BUSY/READY" button is on.

Depress PK2 to enter CMS Bootstrap Mode (see below). The various states, including Initial State and Bootstrap Mode, are shown in figure 8-1.

Possible errors in power-on sequence are given below, System Load Errors.

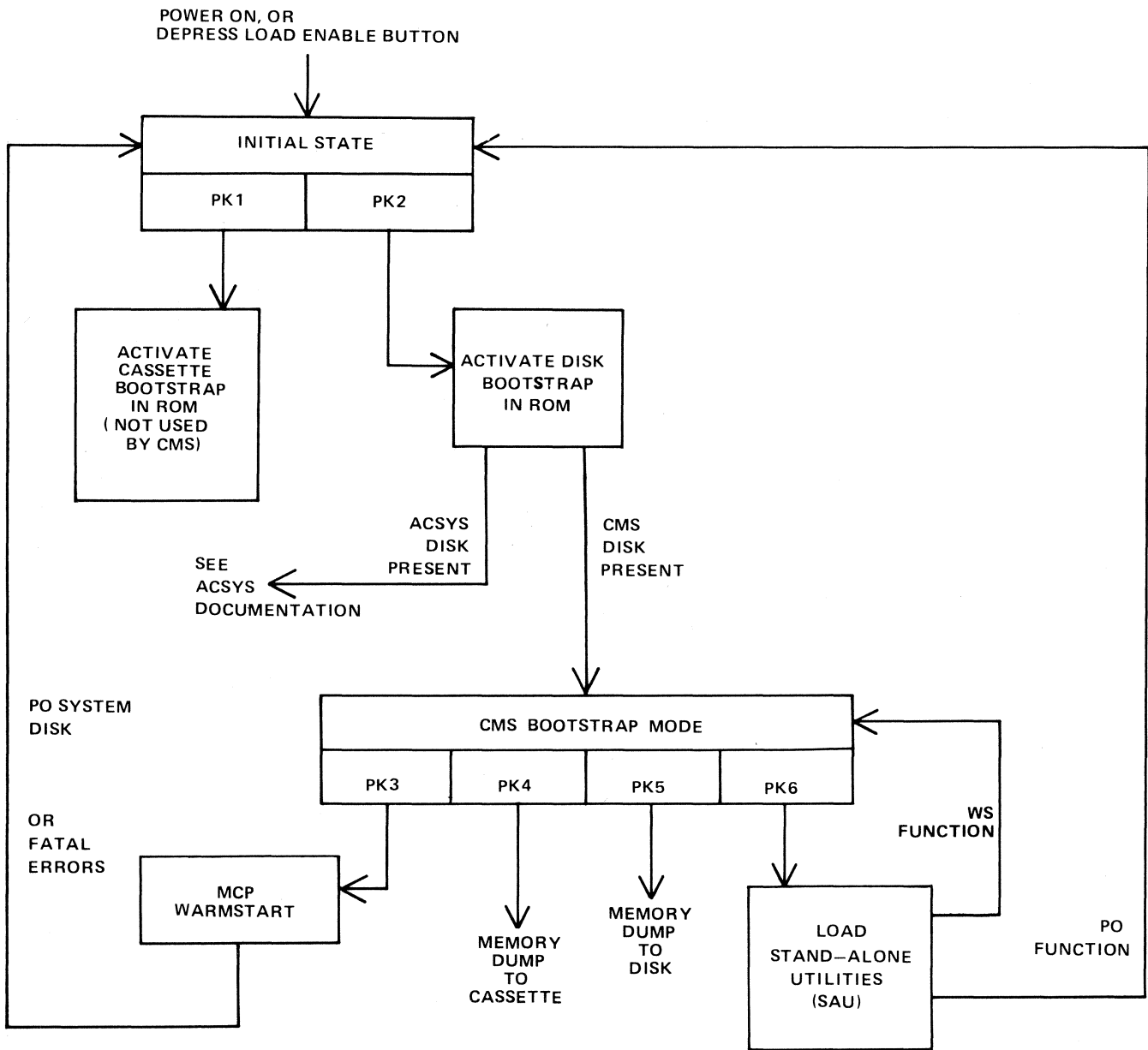


Figure 8-1. B 90 Coldstart and Warmstart

## CMS BOOTSTRAP MODE

From the Initial State, depression of PK2 initiates a ROM load routine which searches through all present disks until it finds the bootstrap code.

Successful load of CMS bootstrap: PK3 through PK6 will be lit, to provide the following facilities:

- PK3 – warmstart MCPX
- PK4 – dump contents of memory to cassette.
- PK5 – dump contents of memory to disk.
- PK6 – enter stand-alone utilities (SAU).

Note:

The search for a bootstrap code proceeds as follows:

- bottom drive of highest-channel disk unit
- top drive of highest-channel disk unit
- ...
- ...
- bottom drive of lowest-channel disk unit
- top drive of lowest-channel disk unit

and stops at the first bootstrap, whether a CMS bootstrap or ACSYS bootstrap. For example, if there is one cabinet with two cartridges, DKA and DKB, then DKB will be searched first. If there is more than one disk cabinet, you can find the channel address from the field engineer.

### A Note on Forcing System Initialization

When the system hangs (that is, it is not performing any functions or responding to any input from an operator, but has not returned to the initial state), it is necessary to force the system to initialize.

This is done by depressing the Load Enable button in the main cabinet. Never switch the cabinet off, or unload disks or cassettes in use, as this can cause media corruption of various kinds.

If the correct procedure is followed, then although disk or cassette files may be only partially created or updated, the system when recovered should be still able to access the media.

## STAND ALONE UTILITIES

The SAU process has the necessary functions to prepare a disk for use on the B 90. This process is used to initially transfer system software (MCP, interpreters, compilers and utilities) to a new disk at a new installation, or to install a new level of MCP at an existing installation. It is also used to condition new disks (IN Initialize) or recondition (RF reformat) existing disks for copying on the B 90.

The SAU functions are brought into memory from disk and operate independent of the MCP.

### Loading Stand-Alone Utilities

The system must be in the CMS Bootstrap state. If the system is in the initial state, depress PK2 to get to CMS Bootstrap State (PKs 3 through 6 lit).

Depress PK6. System will search disk (from which bootstrap was loaded) for a file called SAU.

If this disk does not contain SAU, then any disks (containing bootstrap files) located on a lower channel will also be searched. For each disk you must depress PK6 until search is complete.

(For failures in search see below System Load Errors).

Verify by a printed message that the SAU has been loaded into memory from disk.

```
STAND-ALONE UTILITY
VERSION n.nn.nn
REQUEST "HELP" FOR FUNCTION SUMMARY
FUNCTION
```

### Functions Available

```
initialize a disk: IN
reformat a disk to the initial state: RF disk-name
load files to disk from cassette: LD disk-name FROM cassette-name
copy files from disk to disk: COPY disk-name/file-name TO disk-name/file-name
remove disk files: RM disk-name/file-name
list the disk files and their sizes: LS disk-name/file-name
relabel a disk: RL disk-name
list status of drives: OL
warmstart the MCP: WS
initialize a disk for MTR use: FE (field engineers' use)
terminate stand-alone utility execution: PO
change file-name on disk: CH disk-name/file-name TO disk-name/file-name
clean BSM disk drive read/write heads: CLEAN
compare files on disks: COMPARE disk-name/file-name WITH disk-name/file-name
print disk directories: PDX
duplicate BSMII: DISCOPY drive TO drive
```

When the Stand-Alone Utilities are running, only the ON light is lit. Each of the utilities is initiated by a command on the Alpha Keyboard. Only one utility may run at a time and keyboard input is valid only when no utility is running. Keyboard input entered while a utility is running will be lost except for the first four characters. The reset key will clear all keyboard input since the last OCK key, and any OCK key will terminate keyboard input.

## Common SAU Output Messages

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
FUNCTION ABORTED	Hardware error.	If this message is preceded by another message then follow that message, else notify field engineer.
Dxy DEVICE ERROR DEVICE INOPERABLE	Specified drive was accidentally made Not Ready; or drive unit malfunction.	Ready drive; or notify field engineer.
Dxy DEVICE ERROR OFF CYLINDER	Drive unit malfunction.	Notify field engineer.
Dxy DEVICE ERROR WRITE INHIBITED	Disk in specified drive has its write lockout indicator set.	Ensure that the Write Lockout Hole (BSMD) is covered, or for cartridge - the Write Lockout Plug is flush with surface of cartridge.
Dxy DEVICE ERROR SEEK TIMEOUT	Drive unit malfunction	Notify field engineer.
Dxy DEVICE ERROR CONTROLLER PROBLEM	Drive unit malfunction	Notify field engineer.
FUNCTION	Prompt to request Operator to enter next desired function.	Enter next function, or "HELP" to display list of functions.

Note:

x identifies type of disk

(M = BSMD, F = FIXED, K = CARTRIDGE, S = BSMDII)

y identifies unit

(A, B, C, etc.)

## Disk I/O Errors during SAU

These errors are identified by the messages:

O/P ERROR, I/P ERROR, <unit> DIRECTORY I/O  
ERROR, <unit> DEVICE ERROR

which may be encountered while running the Stand Alone Utilities.

Disk I/O errors indicate a failure to read from (I/P error) or write (O/P error) disk. Such problems should not be allowed to persist on a disk which is to be used to store important information, especially where the

disk is to be used as a systems disk. Therefore an explanation or fix is required before the drive and disk can be considered acceptable for live use. Even if the Stand Alone Utility continues to run satisfactorily, there may be some form of disk corruption. After any of these errors, the media involved should be checked for any corruption which might cause future system problems (for example, the CHECK.DISK, KA, or DA utilities under MCP control).

## A Note On Dual Pack Files

A “dual pack file” is a file which resides on two separate disks or logically identified disks (for example, DFA and DFB).

A dual pack file consists of:

A disk file header on each of the two disks.

At least one and at most sixteen file areas, each of which may be allocated on either disk.

Both parts have the same file name.

Under MCP control, the file may be opened only if both parts are present.

Each file header contains a reference (the pack-id) to the other disk.

Therefore if for any reason one part of the dual pack file is lost, or if the pack-id of one of the disk volumes is changed, the file will be inaccessible under MCP control.

Therefore caution must be exercised when using SAU to initialize, reformat, or relabel any disk containing part of a dual pack file. Dual pack files may be located with the LS function.

In addition to the file-name and area occupied, the LS function will give, for each dual pack file, and overflow pack-id and the total overflow area.

If one part of a dual pack file is lost for any reason, the SAU RM function may be used to remove the remaining part.

The terms “master file” and “overflow file” are sometimes used to distinguish the two parts of a dual pack file, however it should be borne in mind that either part of the file may be regarded as the master file. In this section the term “master” file is used to indicate the part of the file mentioned in a COPY or RM command.

## CH (Change disk file name)

This function allows the operator to change the name of disk files.

Format:

```
CH disk-name/ file-name
                or
                group-name TO disk-name/ file-name
                                or
                                group-name
```

Examples:

To change a file AR030 on a disk ARDISK to PR030X:

```
CH ARDISK/AR030 TO ARDISK/PR030X
```

To change a group of files starting with GL on a disk GLDISK to a group starting with AP:

```
CH GLDISK/GL= TO GLDISK/AP=
```

Output Messages:

MESSAGE	POSSIBLE CAUSE	SUGGESTED ACTION
file-name CHANGED TO file-name	function successful	none
file-name NOT CHANGED ILLEGAL FILE-ID	file-name or group-name contains an illegal character	Check input and re-enter
file-name NOT CHANGED O/P LENGTH OVERFLOW	The resulting file-name is longer than 12 characters (possibly during a group change)	Check the group-name and re-input
file-name NOT CHANGED O/P LENGTH UNDERFLOW	When changing a group name to a smaller group name, the resulting file has no characters (for example, changing F= to = when a file called "F" is present)	Correct the group-name and re-input if necessary; or enter another CH for the file not changed
DUPLICATE FILE file-name	An attempt was made to change a file to the name of an already-existing file	Correct the input and re-enter
file-name NOT FOUND - file-name NOT CHANGED INCOMPLETE DUAL PACK FILE	The file is a dual-pack file and the overflow pack is mounted but the file is not present	Investigate why the other part of the file is missing



MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
PACK disk-name NOT ON LINE - file-name NOT CHANGED - INCOMPLETE DUAL PACK FILE	The file is a dual pack file but the overflow pack is not present	Make the overflow pack present and re-input
file-name NOT CHANGED DUPLICATE FILE ON OVERFLOW PACK	An attempt was made to change the name of a dual-pack file to the name of a file already present on the overflow pack	Correct the input and re-enter

## CLEAN (Clean BSM Drive Read/Write Heads)

The read/write heads of the one megabyte mini drive are cleaned by this function. This cannot be used on any other type of drive. Each head is cleaned in turn by the Burroughs head cleaning diskette.

The procedure the system follows to clean one head is as follows:

Load head onto cleaning surface.

Sequentially access the disk from the outermost track to the innermost.

Sweep heads from the outer to the inner-track and back, ten times.

Unload head.

The current position of the head on the disk is visually displayed by illuminated PK lights.

Example:

CLEAN

Output Messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
DRIVE	Prompt requesting operator to insert cleaning disk into proper drive.	Place cleaning disk into proper drive. Make certain disk is write inhibited (write inhibit hole is covered and is positioned at top corner of disk).
LOAD WRITE INHIBITED CLEANING DISK AND TRANSMIT "OK" TO CONTINUE	Cleaning disk was inserted into drive.	Enter "OK" plus OCK1 to start cleaning procedure; or press OCK1 to terminate the function.
HEAD 1 CLEANED. FLIP WRITE INHIBITED CLEANING DISK AND TRANSMIT "OK" TO CONTINUE.	Head 1 has been cleaned.	Turn disk upside down (write inhibit hole to bottom) so that cleaning surface faces other side; then enter "OK" and press OCK1 to begin cleaning functions.
HEAD 0 CLEANED END CLEAN	Both heads have been cleaned. Function ends.	Remove cleaning disk and store.

Any attempt to clean head 0 before head 1 will result in the "LOAD/FLIPDISK" message being prompted.

### WARNING

Any attempt to use a write-enabled disk or any disk other than the cleaning disk during execution of this function will result in serious damage to the disk drive.

## COMPARE (Compare two disk files)

This function allows the operator to compare files present on disk.

Format:

```
COMPARE  disk-name/  file-name  WITH  disk-name/  file-name
           or          or
           group-name  group-name
```

Examples:

To compare a file PR200 on disk PRDISK with another file PR200 on disk USER:

```
COMPARE PRDISK/PR200 WITH USER/PR200
```

To compare a group of files starting with IN on the disk INDISK with group of files starting with XY on the disk XYZ:

```
COMPARE INDISK/IN= WITH XYZ/XY=
```

Before the files are compared a check is made that the file sizes and number of areas are consistent. If any conflicts are found here, the function will be terminated.

Output Messages:

MESSAGE	POSSIBLE CAUSE	SUGGESTED ACTION
file-name NOT FOUND	Self-explanatory	Check input
FILE SIZE CONFLICTS - END COMPARE	One file contains more records than the other	none
CONFICTIONS IN NUMBER OF AREAS - END COMPARE	The number of areas assigned to the two files are different	none
disk-name/file-name COMPARED WITH disk-name/file-name - NO CONFLICTS FOUND	successful file comparison	none
disk-name/file-name COMPARED WITH disk-name/file - CONFLICTS FOUND	The compare was unsuccessful	Check SPO and error logs for any errors during copying of either file; re-copy file(s) from backup disks and re-run the COMPARE function
PACK disk-name NOT ON LINE - file-name NOT COMPARED - INCOMPLETE DUAL PACK FILE	Comparison of dual pack file was required and the overflow pack was not on line	Make the overflow pack present and re-input COMPARE function

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
file-name NOT FOUND file-name NOT COMPARED INCOMPLETE DUAL PACK FILE	Comparison of dual pack file was required and the overflow pack was present but did not contain the required file	Investigate why the file was not present on the overflow pack
END COMPARE	end-of-job of this S.A.U function	none

## COPY (Copy files disk to disk)

This function allows the operator to copy files from one disk to another. Overflow files will not be copied.

Format:

```
COPY  disk-name/  file-name  TO  disk-name/  file-name
          or      or
          group-name  group-name
```

Examples:

To copy a single file:

```
COPY PR1/PR200 TO PRBU/PR200
```

To copy a group of files:

```
COPY PR1/PR= TO PRBU/PR=
```

To copy all files on one disk to another:

```
COPY PR1/= TO PRBU/=
```

To copy a file and change its name:

```
COPY PR1/PR200 TO PRBU/PR200B
```

Output Messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
DUPLICATE FILE NAME	- System detected a file name on disk identical to that which the operator is attempting to copy. No two files with the same name may reside on a disk. Therefore specified file was not copied.	Normally, remove the existing file with RM; reattempt the COPY. Check input for correct specification of file names.
file-name TOO LARGE	Insufficient available space on destination disk for the specified file to be copied.	Use a different disk, or first remove some files using the RM function.
NAME LIST FULL	When this disk was initialized, the disk directory was constructed to contain a fixed number of file names. The directory has now reached its capacity.	The COPY function ends automatically. IF possible, remove with RM any unwanted file(s) to make room in the disk directory. Then reattempt to COPY the file which encountered the error;

(cont'd)

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
		Replace the disk with a disk that has sufficient room in the disk directory.
Dxy DIRECTORY I/O ERROR	A read or write error was encountered while the system was attempting to access the directory of the specified disk. The directory structure of the input (source) disk may be corrupted.	1. Change the position of the disks and retry. 2. For a group copy, some files may be recovered using single file copy. If these attempts fail then type of error should be analyzed with DA utility and disk re-initialized before re-use.
D/P DISK IS NOT WRITE PERMIT.	Destination disk (disk to which system is copying is write inhibited.	For cartridges: ascertain that write lockout plug is flush with outer surface of cartridge; For BSM: ascertain that the write lockout hole is covered; Retry the COPY.
file-name NOT FOUND	Specified file-name is not on disk.	Check input - re-enter if necessary; Check for correct disk.
PACK disk-name NOT ON LINE	Specified disk is not on-line to the computer.	Check input - re-enter if necessary; Check for correct disk.
INVALID REQUEST	Typing error.	Check input and re-enter.
END COPY	Copy successful (EOJ); Specified group name to be copied is not on disk.	None if copy successful; Check input - re-enter if necessary; Check for correct disk.
file-name COPIED	COPY successful	NONE
CANNOT ALLOCATE AREAS FOR - file-name	Insufficient available space on disk to which operator is attempting to copy this file.	Using KA utility, analyze disk available space.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
file-name NOT CHANGED - O/P LENGTH OVERFLOW	The resulting file-name is longer than 12 characters.	Correct the group name and re-input.
file-name NOT CHANGED - O/P LENGTH UNDERFLOW	The resulting file-name has no characters.	Correct the group name and re-input if necessary.
O/P ERROR - file-name	Error encountered while system was attempting to write specified file to disk.	Allow COPY to continue copying any remaining files. Then reattempt copying the file in which the O/P error occurred.
I/P ERROR - file-name	System encountered errors in the file from which it was reading.	Allow COPY to continue. Then use back-up copy of the specified file and re-attempt the COPY; switch the disks to the opposite drives.
file-name NOT COPIED: ILLEGAL FILE-ID	Specified file-name contains an illegal character (that is, / ?), therefore file will not be copied. Legal characters are 0-9, A-Z, . (dot), - (dash).	Check input and re-enter if necessary.
file-name COPIED: ILLEGAL FILE-ID	Disk-to-disk copy is being made and this warning is given that the new file is copied but with an invalid file-name.	

## Dual Pack Files

The copy function cannot create a dual pack file, however a complete dual pack file may be copied to a single disk.

The master file will be copied first. COPY will then look for a matching overflow file on the overflow pack. If a matching overflow file is found it will be copied and the function will terminate with the message:

file-name COPIED FROM disk-name-1 AND disk-name-2

If the overflow pack is not on line or the overflow file is not found or an overflow file is found which does not match the master file, the function will print the following message, and wait for operator input:

disk-name-file-name IS AN INCOMPLETE DUAL PACK FILE

A MATCHING OVERFLOW FILE ON disk-name-2 IS NOT PRESENT.

PLEASE TAKE ONE OF THE FOLLOWING ACTIONS:

- A. SUPPLY THE CORRECT OVERFLOW PACK AND  
TYPE "A" TO TRY AGAIN.
- B. TYPE "B" TO SKIP THIS FILE.

At this stage the master file pack may be removed if necessary.

The operator should either power on the correct overflow pack and type "A" followed by OCK1 or type "B" followed by OCK1.

If the "A" option is selected the function will repeat its search for the overflow file as above and either terminate normally or repeat the prompt to the operator.

If option "B" is selected the output file will be purged and the function will terminate with the message:  
file-name NOT COPIED, PART OF A DUAL PACK FILE

WARNING

If a level of SAU earlier than 3.00 is used to copy a dual pack file or a family containing a dual pack file, the output disk is liable to be seriously corrupted and to require re-initializing.



## DISCOPY (Duplicate a BSMII Disk)

This function allows the operator to copy the contents of a Burroughs Super Minidisk II (BSMDII) to another BSMDII.

Format:

DISCOPY D5X TO D5X

where X = A, B, etc.

All information is copied from one BSMII to another, that is, disk label, bootstrap, directories and data. Thus a complete disk can be duplicated for backup purposes without initialization. The automatic relocation of bad sectors allows the duplicated directories to remain valid since all information on the duplicated disk will have the same logical, if not physical, address as that on the master disk.

Output Messages:

MESSAGE	POSSIBLE CAUSE	SUGGESTED ACTION
END DISCOPY	end of function	none
NON BSMII DISK SPECIFIED	Either of the disks specified is not a BSMII disk	This function can only be performed on BSMII disks: use COPY for other kinds of disk, including BSM (1MB) disk
PACK	This is a prompt for the operator to give the seven-character disk-name of the input disk	Enter up to seven characters
drive-mnemonic PURGED	An input error has occurred during the copy. The output disk is purged by overwriting track zero with "null" characters.	Investigate cause of error on the input disk, then re-run the DISCOPY function: the purged output disk may be re-used.
FUNCTION ABORTED - drive-mnemonic NOT PURGED	An I/O error has occurred on the output drive during the copy, and the copy is stopped.	Use the LS function to determine which files are successfully copied to the output disk: these may be recovered. Otherwise re-run the DISCOPY function using a different output disk.
drive-mnemonic COPIED TO drive-mnemonic	function successful	none

## FE (Initialize MTR Disk)

A virgin or a formatted disk is initialized to CMS format with suitable sectors reserved for MTR test routines. (These sectors will be denoted as BAD sectors on a KA map of the disk). The surface is checked by writing and reading test patterns to each sector. Bad sectors and the MTR tracks are made unavailable. A disk label is written and the file directory is created with a single, SYSMEM, entry. Sectors 1 through 31 are loaded from a file called "CMSBOOTxxxxx" which can be contained by any on-line disk or by any cassette labelled "SYSB90" which has "CMSBOOTxxxxx" as its first file.

The "xxxxx" characters are ignored by the utility; only the seven leading characters are compared when the on-line disks are searched (that is, the file specifications searched for is equivalent to CMSBOOT=).

In the case of fixed disk, the CMS bootstrap will be loaded from CMSBOOTxxxxx. All other disks initialized by FE will extract the MTR bootstrap.

The MTR tracks will not be included in the number of bad sectors, but if the disk is reformatted (see RF), the total number of sectors removed from the available table will be referred to as BAD.

Example:

FE

Output Messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
DRIVE	Prompt allowing entry of drive where disk to be initialized is located.	Enter Drive (ex: DKA, DMB, etc)
DATE	Prompt allowing entry of initialization date.	Enter date (format MM/DD/YY)
FILES	Prompt allowing entry of maximum number of files for this disk.	Enter maximum number of files this disk will contain (less than 2805).
SERIAL	Prompt allowing entry of disk serial number.	Enter serial number (6 numerals).
PACK	Prompt allowing entry of disk name.	Enter disk-name (up to 7 legal characters) Legal characters include 0-9, A-Z, . (dot), - (dash).
OWNER	Prompt allowing entry of user-defined information.	Enter valid information (up to 14 characters).

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
RESTRICTED	Prompt requiring response of "N" at this time.	Enter "V".
BOOTSTRAP VERSION n.nn.nn USED END FE	FE (EOJ)	None.
NO OF BAD SECTORS number	Disk was successfully initialized. If any bad sectors were detected the number is displayed.	None.
TRACK 0 BAD	Sector(s) in Track 0 are bad and cannot be used. FE will end, as a CMS disk must contain reserved information in Track 0 (ex: warmstart bootstrap).	Select another disk to be initialized with FE.
SYS90 NOT PRESENT LOAD CMS BOOTSTRAP AND HIT JCK1 TO CONTINUE	System failed to find a bootstrap file from either disk or cassette	Load another drive with a disk containing CMSBOOT or load a cassette with name SYS90 containing CMSBOOT. Then press JCK1. System will re-attempt search for CMSBOOT.
BAD MTR TRACK	Any MTR track contains a bad sector.	
CYL xxx BAD	In cases where certain cylinders are used for MTR purposes but not removed from the available table discovery of a bad sector will terminate the function.	Select another disk to be initialized by FE.

## IN (Initialize a Disk)

The MCP (Master Control Program) requires that any disk to be used on the system have a valid CMS disk label, disk directory, and available area table. In addition, each sector of the disk must be initialized with its address.

The IN function performs this disk initialization. It will check the recording surface of each disk by writing and reading test patterns to each sector of the disk. Any "bad sectors" (unusable or unreadable) will be removed from the disk's available area table, and the number of bad sectors will be displayed to allow badly worn disks to be discarded.

The function will also write a disk label containing information supplied by the operator to the appropriate prompts, below, and create a disk directory of the appropriate size required for the number of files specified, plus a single, SYSMEM, entry. Sectors 1 through 31 of the disk are loaded from a file "CMSBOOTxxxxx" which may be located on any on-line disk or on any cassette labelled "SYSB90" which has "CMSBOOTxxxxx" as its first file. Before loading can take place, the correct identification string must be found in sector one of this file, namely, B90DISKINITVAOF. The disks are searched in descending order, cartridge disks first, then a further scan of Burroughs Super Mini Disks.

Since all fixed disks must be suitable for MTR purposes, the functions IN and FE are identical in this particular case. The relevant MTR tracks will be checked and/or removed and the CMS bootstrap is written.

Example:

IN

Output Messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
NO OF BAD SECTORS number	Disk was successfully initialized. If any bad sectors were detected the number found is displayed.	None.
BOOTSTRAP VERSION n.nn.nn USE) END IN	IN successful (EOJ)	None.
SYSB90 NOT PRESENT LOAD CMS BOOTSTRAP FILE AND HIT OCK1 TO CONTINUE.	System failed to find bootstrap file on disk or cassette.	Load another drive with a disk containing CMSBOOT or load a cassette with name SYSB90 containing CMSBOOT as its first file. Then depress cassette, then press OCK1. System will re-attempt search for CMSBOOT.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
TRACK 0 BAD	Sector(s) in Track 0 are bad and cannot be used. IN will terminate, as a CMS disk must contain reserved information in track 0 (ex: warmstart bootstrap).	Select another disk to be initialized.
DRIVE	Prompt allowing entry of drive in which disk to be initialized is located.	Enter drive (ex: DKB, DKA, etc.)
DATE	Prompt allowing entry of initialization date.	Enter date (format MM/DD/YY for month, day, year; e.g. 01/23/79)
FILES	Prompt allowing entry of maximum number of files for this disk.	Enter number of files disk will contain (less than 2805).
SERIAL	Prompt allowing entry of disk serial number.	Enter serial number (6 numerals).
PACK	Prompt allowing entry of disk name.	Enter disk-name (up to 7 characters).
OWNER	Prompt allowing entry of user-defined information.	Enter any valid information (up to 14 characters).
RESTRICTED	Prompt requiring response of "N" at this time.	Enter "N".

## LD (Load Disk)

This function allows the operator to load all files from a dump tape to a disk. A dump tape is one produced by the DUMP or UNLOAD functions of the utility "LD" (which run under MCP control). Each sector of data written to disk is verified. The SAU LD function cannot create a dual pack file.

Format:

LD disk-name FROM library-tape-name

Example:

LD ARDISK2 FROM ARTAPE

Output Messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
file-name LOADED	Displayed for each file loaded and verified.	None.
LOAD REEL number	Dump tape is a reel from a multi-reel dump and the next reel is required.	Supply next reel.
END LD	LD (EOJ)	None.
NOT DUMP TAPE	Specified tape is not a correctly formatted dump tape. Function ENDS.	Supply correctly formatted dump tape; re-initiate LD.
multi-file-name NOT PRESENT. LOAD CASSETTE AND HIT OCK1 OR HIT ANY KEY THEN OCK1 TO TERMINATE.	Specified tape is not installed and ready; Function ends.	Install and ready tape; re-initiate LD; or press OCK1 to retry search for multi-reel name tape; or press any key plus OCK1 to terminate LD.
UNRECOVERABLE CASSETTE ERROR	Error was encountered while system was attempting to read cassette. Normally caused by accidentally opening cassette drive unit. Function ends.	Ready cassette drive unit; re-initiate LD.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
PACK disk-name NOT FOUND	Specified disk is not ready; specified disk is not on-line to computer.	Ready the disk; Check for correct disk.
O/P DISK NOT WRITE PERMIT	Disk to which files are to be loaded is write protected. Function ends.	Ensure that write lockout hole is covered (for BSMD), for cartridge - ensure write lockout plug is flush with surface of disk cartridge; re-initiate LD.
DUPLICATE FILE NAME - file-name	File of specified name already exists on destination disk. A disk may not contain 2 files with the same name. LD continues loading other files, ignoring any duplicates.	Remove (with RM) the existing file from disk, then re-initiate LD.
CANNOT ALLOCATE AREAS FOR - file-name.	No appropriately sized available area on the destination disk for specified file.	If specified file is desired, replace this disk with a disk having more available area. Then re-initiate LD.
O/P ERROR - file-name	Disk write error encountered while system was loading specified file.	Place disk in opposite drive and re-initiate LD.
AREA SIZES TOO SMALL FOR - file-name.	Insufficient available area on disk for specified multi-area file.	If specified file is desired, replace this disk with a disk having more available area. Then re-initiate LD.
NAME LIST FULL	No space remaining in disk directory for another file name. Function ends.	Use another disk, or remove unwanted files with the RM function.

## LS (List File Sizes)

This function allows the operator to print name and sizes in sectors of files on a specified disk. For dual pack files, both portions of the files and their sizes are printed.

Format:

```
LS disk-name/ file-name  
                or  
                group-name
```

Example:

To list file sizes of all the files on the disk ARDISK2

```
LS ARDISK2/=
```

Output Messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
PACK disk-name NOT ON LINE	Specified disk is not on line to the computer.	Check input - re-enter if necessary; Check for correct disk.
END LS	LS successful (EOJ)	None.
Dxy DIRECTORY I/O ERROR	A read or write error was encountered while the system was attempting to access the directory of the specified disk. The directory structure of the disk may be corrupted.	



## OL (Print Status of Drives)

This function allows the operator to print the status of all cassette and disk drives.

Example:

OL

Output Messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
END OL	OL successful (EOJ);	None.
disk drive disk-name	Specified disk is resident in specified drive.	None.
disk drive NOT READY	There is no disk in this drive; the disk in the drive has not been set to run; the disk in the drive has not come up to proper speed; system does not recognize disk.	If applicable - set disk to run; or allow disk to attain proper speed; make certain disk has a proper "label".
disk drive NOT CMS - STANDARD	Disk in this drive does not have a valid CMS label.	Disk must be initialized with IV to create valid CMS label.
cassette drive NOT READY	There is no cassette in this drive; cassette has not been loaded into drive properly;	Check for proper loading of cassette in drive.
cassette drive UNLABELLED	Cassette in specified drive does not have a valid label.	Use PG to create valid cassette label.
cassette drive multi-file-name file-name	Cassette in this drive has the specified name.	None.
disk-drive TEMPORARILY NOT AVAILABLE	The door of the disk drive has been opened.	Close the door.

MTR cassettes do not have labels corresponding to the correct CMS format and thus will appear "UNLABELLED".

## PDX (Print Disk Directories)

This function allows the operator to print the disk directories, disk label and any sector in hexadecimal.

Format:

PDX

The function operates in interactive mode. The operator is prompted to supply the input.

Output Messages:

MESSAGE	POSSIBLE CAUSE	SUGGESTED ACTION
DRIVE	This is a prompt to the operator	Enter the device mnemonic of the disk to be read; for example, DKA
IS DIRECTORY REQUIRED <Y OR N>	This is a prompt to the operator	Enter "Y" if a complete directory print is required, otherwise enter "N"
SECTOR <4 character hex value or null>	The operator is prompted to provide the hex address of a sector to be displayed. This prompt is repeated until a null response (just an OCK key) is given.	Enter the 4-character hex sector address followed by OCK1, or just OCK1 to terminate the function
COMPLETE <Y OR N>	The operator is prompted to state if printing of all sectors in the disk directory is required or only the first sector of each entry	Enter "Y" or "N" as appropriate
LABEL CORRUPT	The check-string "SL9INTERNL" in sector zero has not been found	none: use backup copy of disk for normal use
END PDX	end of PDX function	enter dsired S.A.U function

## **PO (Power Off)**

This function allows the operator to terminate the execution of the Stand-Alone Utilities.

Example:

PO

The utility displays the message

\* \* \* END STAND ALONE UTILITY \* \* \*

and causes the B 90 to return to the initial state (PK1 and PK2 lit, refer to figure 11-1).

## RF (Reformat Disk)

This function provides all the capabilities of IN (initialize), except the disk recording surface test. A CMS label and a CMS disk directory are written to the disk. Any information previously contained on the disk will be lost. The disk label will contain information supplied by the operator's responses to the appropriate prompts, below, and the directory will be of the minimum size required for the number of files specified. The original contents of the disk label will be displayed to record the change and to assist re-input of the same data when required. The bootstrap is written back to track "0" in the same manner as for IN.

Format:

RF disk-name

Example:

RF ARDISK2

Output Messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
DATE	Prompt allowing entry of RF date.	Enter date (format MM/DD/YY)
FILES	Prompt allowing entry of maximum number of files disk will contain.	Enter number of files disk will contain (less than 2805).
SERIAL	Prompt allowing entry of disk serial number.	Enter serial number (6 numerals).
PACK	Prompt allowing entry of disk name.	Enter disk-name (up to 7 characters).
OWNER	Prompt allowing entry of user-defined information.	Enter any valid information (up to 14 characters).
RESTRICTED	Prompt, requiring a response of "N" at this time.	Enter "V".
BOOTSTRAP VERSION n.nn.nn USE) END RF	RF End of Job (EOJ)	None.
PACK disk-name NOT ON LINE	Specified disk-name is not on-line to computer.	Check input (re-input if necessary); Check for correct disk).

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
Dxy DIRECTORY I/O ERROR	A read or write error was encountered while system was attempting to access directory of this disk. Directory of this disk may be corrupted.	Switch disk to alternate drive and re-initiate RF; or
SYSB 90 NOT PRESENT. LOAD CMS BOOTSTRAP AND HIT OCK1 TO CONTINUE.	System failed to find a bootstrap file from either disk or cassette.	Load another drive with disk containing C4SB00T, or load a cassette with name SYSB90 containing C4SB00T as its first file. Then press JCK1. System will re-attempt search for C4SB00T.

## RL (Relabel a disk)

This function allows the operator to change a disk's name without affecting the remaining contents of the disk.

Format:

RL disk-name

Example:

RL AP2

Output Messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
END RL	RL successful (EOJ).	None.
PACK	Prompt to request operator to enter the new disk name.	Enter new disk name (up to 7 legal characters).
PACK disk-name NOT ON LINE	Specified disk is not on line to the computer.	Check input - re-enter if necessary; Check for correct disk.
Dxy DIRECTORY I/O ERROR	A read or write error was encountered while the system was attempting to access the directory of the specified disk. The directory structure of the disk may be corrupted.	

Warning:

If RL is used on a disk containing part of a dual pack file, that file will be inaccessible under MCP control.

There are two ways to prevent this from occurring:

- Remove the file before relabelling the disk, or
- Copy the entire file to a single pack file before relabelling.

There are two ways to resolve the problem if it does occur:

- Use Stand-Alone Utility (RM) to remove the file, or
- Use Stand-Alone Utility (RL) to relabel the disk to its original name.

## RM (Remove Disk Files)

This function allows the operator to remove files from disk.

Format:

```
RM disk-name/ file-name
                or
                group-name
```

Examples:

To remove a single file:

```
RM PR1/PR200
```

To remove a group of files:

```
RM PR1/PR=
```

To remove all files from disk:

```
RM PR1/=
```

Unlike the RM utility that runs under MCP control, the Stand-Alone RM will remove system software (MCPX, COBOLINTX, BILINTERPX, etc.) without a warning message, the same as it removes other programs.

Output Messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
file-name REMOVED	RM successful.	None.
PACK disk-name NOT ON LINE	Specified disk is not on-line to the computer.	Check input - re-enter if necessary; Check for correct disk.
file-name NOT FOUND	Specified file-name is not on disk.	Check input - re-enter if necessary; Check for correct disk.
END RM	RM successful (EOJ); Specified group name to be removed is not on disk.	None if RM successful. Check input - re-enter if necessary. Check for correct disk.

## Dual Pack Files

If a file to be removed has an overflow area or another disk volume:

- if both parts of the file are available, both are removed and the following message is printed:

```
file-name REMOVED FROM disk-name-1 AND disk-name-2
```

- if the overflow file is not found for any reason or if a file is found with the same file-id but which does not match the master file, then the following message is printed:

disk-name/file-name IS AN INCOMPLETE DUAL PACK FILE  
A MATCHING OVERFLOW FILE ON disk-name-2  
IS NOT PRESENT. PLEASE TAKE ONE OF THE  
FOLLOWING ACTIONS:

- A) SUPPLY THE CORRECT OVERFLOW PACK  
AND TYPE "A" TO TRY AGAIN
- B) TYPE "B" TO SKIP THIS FILE
- C) TYPE "C" TO REMOVE THE INCOMPLETE FILE

If option "A" is selected then either the complete file will be removed or the above prompt will be repeated.

If option "B" is selected then the following message is printed:

file-name NOT REMOVED, PART OF DUAL PACK FILE

If option "C" is selected then the following message is printed:

file-name REMOVED ONLY FROM disk-name-1



## **WS (Warm Start)**

This function causes a “branch” to the warmstart routine stored in the Read Only Memory (ROM) of the machine. It will cause the CMS operating system (file-name MCPX) to be loaded from disk into memory.

### **Operating Procedure:**

Type “WS” then press OCK1. Wait for PKs 3, 4, 5, and 6 to be illuminated.

Press PK3.

Enter today’s date in the format:

MM/DD/YY

when prompted to do so.

For further information, refer to the section headed “Loading the MCP”.

## LOADING THE MCP

(This process is also called the “warmstart procedure”).

The B 90 MCP code file is named “MCPX”.

From the CMS Bootstrap Mode (PK3 to PK6 lit), depress PK3.

The bootstrap will search for a disk file called “MCPX”. The search for the MCP proceeds as follows:

bottom drive of highest-channel disk unit

top drive of highest-channel disk unit

....

....

bottom drive of lowest-channel disk unit

top drive of lowest-channel disk unit

For failures in the search, see below, System Load Errors.

If the MCP search is successful, the MCP is loaded to memory and MCP initialization takes place. The activity during this process can be distinguished on the D-lights.

The initial part of the MCP is loaded (D2 light flickers).

The console printer is initialized (if one exists).

AVR (Automatic Volume Recognition, see below) is performed on all peripherals (D2 flickering with D4 and D7 on).

For mini-disks, you will hear the disk start to click.

For cartridge disks, AVR procedures are very quick.

The system will print on the SPO (console or self-scan, according to data in the SYSCONFIG file) an MCP version message, followed by a list of on-line peripherals that are powered up.

### NOTE

The version of the MCP is identified by mark/level/patch numbers. For example, version 3.02.27 is mark 3, level 3.02, patch 3.02.27. A new software release is denoted by a higher level number (for example, 3.03). Within a release, higher patch numbers indicate improved versions of that level. For example, the application of two patches to 3.02.27 will create an MCP version 3.02.29 (see PATCHMAKER for details of how to patch MCP files).

The system will request the date. Depress the Ready Request button and verify that the ON, READY and ALPHA lights are lit.

Enter the date as requested, followed by OCK1. (Leading zeros are optional). The system prints a date message.

The MCP automatically loads the program SYS-SUPERUTL (D2, D3, D4, D5, D6 and D7 on).

If logging is specified in the SYSCONFIG file, the message

“COMMENCING LOG FILE CONSOLIDATION”

is given, and the TL utility is loaded (D2, D3, D4, D5, D6 and D7 on) and executed (D2 flickering with D4 and D6 on). At end of TL execution, the message

“TRANSFER COMPLETED”

“COMMENCING LOG FILE REALLOCATION”  
“LOGGING IS INITIATED ON MM/DD/YY”

is given, and SYS-SUPERUTL creates new log files (D2, D3, D4 and D5 on).

Optionally, (depending on SYSCONFIG), a user program is loaded and executed (D2, D3, D4, D5, D6 and D7 on).

The warmstart is complete and the MCP enters idle state (see below).

It is advisable not to enter system commands until the complete warmstart procedure is over, otherwise confusion can result. Also, such input is not entered in the system log.

Example:

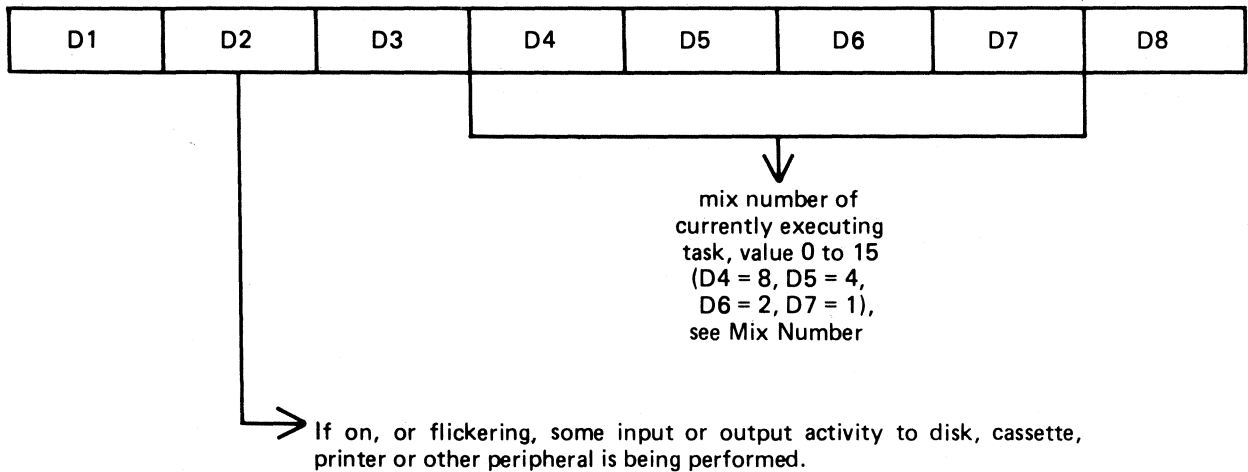
```
B-90 DCMCP VERSION 03.02.32 79313
DFA FA/ 0 FILES OPEN
DFB FB/ SYS DISK 0 FILES OPEN
ENTER DATE AS MM/DD/YY
01/01/80
01 JAN 80 80001 TUE
COMMENCING LOG FILE CONSOLIDATION
TRANSFER COMPLETED
COMMENCING LOG FILE REALLOCATION
LOGGING IS INITIATED ON 01/01/80 (MCP VER
...SION 03.02.32)
```

For possible errors, see System Load Errors, later.

# BASIC OPERATION UNDER MCP CONTROL

## D-Lights

During MCP execution D-lights D1 to D8 give an indication of system activity, as follows:



If the lights D4 to D7 are all off, this indicates system activity not related to a particular program. For example, AVR procedures result in D2, D4 and D7 being lit.

If the lights D4 to D7 are all on (value 15), this indicates processing of an input request or loading of a program.

## MCP States

After warmstart is complete, the MCP is either idle or executing a task. The idle state is identifiable by the absence of activity on the D-lights. In this state, three distinct patterns can be encountered:

D2, D3 only lit:

The last activity was a program gone to EOJ.

D1, D3, D4, D5 only lit:

The last activity was a SYS-SUPERUTL function (IR, LB, LF, PD, RM, KX, or CH).

D1, D3, D4, D5, D6, D7 only lit:

The last activity was a system intrinsic.

When the MCP is executing a task, the mix number is evident in D-lights D4 to D7 (see Mix Numbers).

At any time when the system is idle, it is valid to terminate the MCP by a PO of the system disk (see PO intrinsic).

## Mix Numbers

The following table indicates the mix numbers for B 90 activities:

<u>mix-number</u>	<u>task</u>	<u>D4-D7 lights</u>
0	* system activity	none
1-8	user programs	various
9	* AVR	D4, D7
10,11	utility programs	various
12	+ SYS-SUPERUTL	D4, D5
13	MCS	D4, D5, D7
14	* NDL	D4, D5, D6
15	* program loader/input requests	D4, D5, D6, D7

\* these tasks are not shown in the response to the MX intrinsic.

+ SYS-SUPERUTL is not shown in the response to the MX intrinsic unless a particular function is being performed: in this case, the MX response shows the name of the function, for example, 12/RM, or 12/PD

Normal user programs or compilers are allocated the next available mix numbers in the range of 1-8. Utilities are allocated the lowest available mix number from 10 or 11.

## Automatic Volume Recognition (AVR)

This procedure is carried out if any new media is loaded onto the system. If the procedure fails, the device is made not ready. The procedure varies for different media.

For fixed disk, disk cartridge, Burroughs Super Mini (BSM) disk, Burroughs Super Mini II disk, and ICMD, AVR will attempt to read the label.

For cassette, AVR will search for a CMS label or scratch label, otherwise the cassette is treated as unlabelled.

In all cases, if transient errors are suspected, Ready the peripheral (see RY intrinsics) re-initiate the AVR process.

## Console Keyboard Under MCP Control

Under MCP control, the console keyboard may be used to enter system commands to the MCP, or solicited data to a program. Commands to the MCP may only be entered when the READY light is lit. Data to a program may only be entered on the alpha keyboard if the ALPHA light, but not the READY light, is lit. Input is terminated by an OCK key, or a PK key only if the corresponding PK light is lit. It can be seen that the keyboard can exist in one of three states; inactive, system enabled, or program enabled.

Since an understanding of the operation and behaviour of the keyboard and its indicators is essential, some important points are noted here.

If a disabled key is pressed at any time, or too much data is entered at once, then an error bell will ring and the ERROR indicator will light, this error condition must be reset before any further keyboard input can be made.

The RESET key has two basic functions, resetting the error condition explained above, or clearing the information keyed since an OCK or PK key was last pressed. This key is necessarily enabled whenever any alpha or numeric key is enabled.

When some incomplete information has been keyed either to the MCP or a program then it is necessary to terminate some information to this destination before entry to the other destination can be allowed. For example, once the ready light is lit an OCK key must be pressed to terminate input to the MCP before any input can be keyed to a waiting program. This restriction stands even if the reset key is used.

If the keyboard is enabled for input then the D-lights are not lit.

In order to enable keyboard input to the MCP the ready enable key must be pressed. This key will be ignored if the SCL/LOADER routine of the MCP is currently executing a system command or loading a program (D4, D5, D6, D7 lit) and a wait of up to 30 seconds may be necessary before any keyboard input can be made. Similarly if the MCP is very busy when a request for input is made there may be a delay of one or two seconds before the keyboard will become enabled for input. A maximum of two characters can be entered in this time, or a keyboard error condition will arise.

If the system is being used with the self-scan screen operating as a SPO device as well as a console file, then the ready enable key has an additional function. This key must be depressed in order to prompt the MCP to display either the complete screenful of messages on the screen, or the last message on the bottom line of the screen (scrolling the other information up one line). When the screen is being used as a console file, the screen will display information from the program using it until the ready enable key is depressed.

If the screen is displaying SPO information, depression of an OCK will return the display to the console file information.

## Interrupting the MCP

It is possible to interrupt the MCP while it is running, this will inform the MCP that some action must be taken. The ability to interrupt the MCP is a good indication that the system is running satisfactorily. The only conditions where it may prove impossible to cause an interrupt is when the MCP is processing a system command or loading a program. Typical interrupts and the consequent processing include:

Press READY ENABLE button. This will cause the MCP to enable the keyboard for input.

Opening or closing the serial printer cover will cause the MCP to prevent or allow output to the printer accordingly.

Loading a disk or cassette will cause the MCP to read the label and perform the AVR procedure.

## MEMORY DUMP TO CASSETTE

The system must be in the CMS Bootstrap state. If the system had clear-started, depress PK2 to get to the CMS Bootstrap state (PK3 to PK6 lit).

Depress PK4. The numeric light will be lit.

Insert a write-enabled cassette in any cassette drive unit, and wait until fully rewound.

Depress the numeric key (1 to 4) corresponding to the drive unit containing the cassette (CTA=1 etc.).

The contents of RAM will be written to the cassette. During the dump, an indication of the memory address being dumped is displayed on the PK lights. At the end of the dump, PK17 to PK24 lights are lit. The cassette will be labelled "MEMDUMP/MEMORY". The system will return to the bootstrap state (PK3 to PK6 lit).

Remove the cassette, clearly mark it with the date and time, and submit it with details of the fault to your support personnel.

For possible errors, see System Load Errors, below.

## MEMORY DUMP TO DISK

The system must be in the CMS Bootstrap state. If the system had clear-started, depress PK2 to get to the CMS Bootstrap state (PK3 to PK6 lit).

Depress PK5. The system will search the disk (from which the bootstrap was loaded) for a file called MEMDUMP.

The utility GEN.DUMPFL may be used to create a suitable file on the system disk. This must have been done before a dump to disk is attempted. It is recommended that for minidisk-based systems a spare minidisk is kept with a MEMDUMP file on, and this disk is the only disk loaded when the memory dump to disk is taken. This will avoid any possible confusion between MEMDUMP files on different disks.

The contents of RAM will be written to the MEMDUMP file. At the end of the dump, the CMS Bootstrap state will be entered (PK3 to PK6 lit).

For possible errors, see System Load Errors, below.

### NOTE

Valid memory dumps will be produced only if the pertinent error condition was the last event on the system. Invalid memory dumps will occur if the machine main cabinet has just been switched on, or the error condition did not occur under MCP control, or PK1 was depressed while in initial state (PK1 and 2 lit).



## SYSTEM LOAD ERRORS

The following table of errors cover all symptoms found during start-up of the B 90, warmstart, memory dump to disk or cassette, or entry to Stand-alone utilities.

SYMPTOM	POSSIBLE CAUSES	SUGGESTED ACTION
Sequential lighting and extinguishing of lights does not occur on entry to Initial State (for example, depressing Load Enable button).	MTR switch in wrong position.	Set MTR switch to "normal". Depress Load Enable button.
From Initial State, PK2 is ignored	Keyboard locked in "shift" mode.	Depress shift key. Depress PK2.
Depression of PK2 causes numeric light to be lit.	PK1 was depressed by mistake (this clears memory).	Depress Load Enable. Depress PK2.
Depression of PK2 causes keyboard lights plus ERROR light, to be lit:		
(a) D1 through D8 lit	No bootstrap found (disk not initialized correctly).	Check on disk used. Reload with correct disk. Press Load Enable. Press PK2. See below for diagnosis.
(b) one D light not lit, plus PK lights lit	Disk media or drive fault.	Note D and PK lights. Check that disk is not at fault by using disk in another drive. Power off faulty disk, replace with backup copy. Request technical assistance.
All keyboard indicators flash	Memory parity	Depress Load Enable, depress PK1 (to clear memory), then depress Load Enable again, and retry. Request technical assistance if not successful.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
Warmstart stops with D2 lit.	Console printer not ready.	Check forms transport closed. Check printer cover down. Check levers in correct position. If system does not continue, depress Ready then OCK1.
Print head does not initialize at start of warmstart.	Console printer faulty.	Open and close cover. Repeat warmstart. Request technical assistance.
Memory dump to cassette: System warmstarts	PK3 was depressed instead of PK4.	Memory dump cannot now be taken. Allow warmstart to complete.
Memory dump to cassette: keyboard ERROR light is lit when numeric key depressed.	Wrong cassette drive has been selected, cassette is not write-enabled, or tape is rewinding.	Correct fault in cassette. Depress Reset. Enter correct numeric key.
Memory dump to cassette: ERROR light is lit after 10 seconds.	Cassette is inserted backwards.	Correctly insert cassette. Depress Reset. Enter correct numeric key.
Memory dump to cassette: ERROR light is lit during dump.	Irrecoverable tape error, or accidental opening of cassette drive unit.	Allow cassette to rewind, or replace cassette. Depress Reset. Enter numeric key.
Memory dump to disk: Depression of PK5 causes one D light plus ERROR, to be lit.	No MEMDUMP file found on disk.	Load a disk with a MEMDUMP file and depress PK2 then PK5 to retry the dump.
Memory dump to disk: ERROR light is lit during dump	MEMDUMP file on disk is too small to hold memory contents for this machine.	Load disk with larger MEMDUMP file; depress PK2 then PK5 to retry the dump.
Memory dump to disk: system warmstarts	PK3 was depressed instead of PK5.	Memory dump cannot now be taken. Allow warmstart to complete.

SYMPTOM	POSSIBLE CAUSES	SUGGESTED ACTION
Entry to S.A.U: Depression of PK6 causes one D light, plus ERROR, to be lit.	SAU could not be loaded from disk: disk error.	Note D and PK lights. Remove disk, replace with backup copy. Press Load Enable, PK2 and PK5. See below for diagnosis.
Entry to S.A.U: load stops with all keyboard lights lit.	Console printer not ready.	Check forms transport closed. Check printer cover down. Check levers in correct position. Utility should continue.

## Diagnosis of Disk Errors at System Load Time

The disk load routines may fail for two distinct reasons, both indicated by the D lights.

If all D lights D1 to D8 are lit, then the required information (bootstrap, SAU, MEMDUMP) was not on disk.

If not all D lights are lit, then information found on disk was not reliable due to disk or disk drive errors.

If all D lights D1 to D8 are lit except one, then the failure occurred during bootstrap load and the extinguished D light indicates the disk unit at fault.

If only one D light is lit, then the failure occurred during other disk activity (MCP load, SAU load, memory dump to disk), and the illuminated D light indicates the faulty disk unit.

In addition, the PK lights indicate the nature of the fault.

Make a note of each D light and PK light setting and take action as in the tables above. The meanings are as follows:

D1 channel 0

D2 channel 1

---

D8 channel 7

PK1 off = top drive; on = bottom drive

PK2 off = seek complete

PK3 off = end of cylinder

PK4 off = search complete

PK5 off = more details in PK9 to PK16

PK6 on = operational

PK7 off = seek incomplete

PK8 on = good status

PK9 on = equal

PK10 on = on cylinder

PK11 off = illegal seek

PK12 on = write inhibit

PK13 off = sector not found

PK14 off = LRC error (parity)

PK15 off = illegal command sequence

PK16 off = device error

(PK9 to PK16 are significant only if PK5 is off).

PK17 - PK24 indicate number of attempted retries before  
error is declared.

## ERRORS UNDER MCP CONTROL

The following table of errors covers many symptoms found while using the B 90 under normal control of the MCPX, with suggested causes and actions to take.

SYMPTOM	POSSIBLE CAUSES	SUGGESTED ACTION
Unidentifiable problem	MCP file corrupted	Use backup copy of MCP
Little happening, with D2 on, D3 off most of time	System thrashing; too many jobs or too large jobs in mix.	If console not in use, set CT ON. If light pattern above PK17 to PK20 is : on, on, off, on, then thrashing is confirmed : D lights 4 to 7 indicate mix number of executing task. Try entering MX to clear condition : if not successful, take memory dump and do clear start : request technical assistance.
	Extended memory not correctly used.	Run CONFIGURER and warmstart if necessary.
PK lights 17 to 24 flashing with "DF" pattern (on, on, off, on, on, on, on)	hardware disk I/O error during MCP task	Take memory dump : request technical assistance to analyze memory dump to find disk address for use in fixing the disk. To help find the fault, warmstart and run the KA utility on the system disk before executing any programs. (See KA, section 4). If shows any "AREAS ASSIGNED TWICE" the disk directory may be bad. DO NOT USE THIS DISK until recovery operations have been completed.

SYMPTOM	POSSIBLE CAUSES	SUGGESTED ACTION
PK lights 17 to 24 flashing with "AC" pattern (on,off,on,off,on,on,off,off) or "AD" pattern (on,off,on,off,on,on,off,on)	see below:	(Take recovery action as for "DF" messages above: DO NOT USE THE DISK until recovery action is completed.)
	hardware disk I/O error during MCP disk directory operation.	Request technical assistance to fix the disk.
	logical error in disk directory due to previous fault.	Rerun using backup disk : analyze corrupted disk using S.A.U. PDX function and report.
non-zero retry count on PD of system disk	Deteriorating disk performance.	Ignore if 10 or 20 during normal running in one day. Call field engineer if greater rate than usual rate for the site.
	Improper care of removable disks: possibly failure to re-initialize disks after long use	All frequently used removable disks should be cleaned and re-initialized regularly (for example, on a monthly basis).
Logging not taking place	SYS-SUPERUTL not on systems disk.	Load SYS-SUPERUTL to systems disk.
	TL not on systems disk	Load TL to systems disk
	Logging not specified in SYSCONFIG file.	Rerun CONFIGURER and warmstart.
PD, RM function not available	No SYS-SUPERUTL	Load SYS-SUPERUTL to systems disk.
	SYS-SUPERUTL already performing one of its functions (see discussion in section 4).	Check with MX if SYS-SUPERUTL is performing a function (mix-number 12 gives the name of the function): wait then re-enter command.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
Cannot run data comm	No NDLSYS file on system disk	Load NDLSYS file (you may need to run an NDL compilation.
	NDLSYS file does not correspond to actual network on machine.	Load correct NDLSYS file; or recompile NDL program.
	NDL.INTERP not present on system disk.	Make sure the correct interpreter (for example MICROPOL) is named NDL.INTERP and located on the system disk.
	No MCS program	Ensure that the correct MCS is executing as well as any user data comm program.
Degraded performance	Lack of memory due to incorrect use of Extended memory, wrong SYSCONFIG file on system disk.	Run CONFIGURER, repeat after warmstart.
	Lack of memory due to memory fault at warmstart time.	Call field engineer.
System not active: programs in mix but D-light pattern does not change; keyboard input (for example, MX) is possible.	Program swapped out waiting on memory.	Enter <mix> GO to resume program execution; if symptom persists, program is too large for available memory.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
System not active : programs in mix but D-light pattern does not change; keyboard input (for example, MX) causes error bell to ring	Console file input in process.	Complete input using an OCK key.
	System printer is jammed.	Open and close printer cover to initialize print head. Repeat if not successful first time. If still not successful, perform a warmstart.
	Peripheral (possibly line printer) hang	Enter CL command. If not successful, make per- ipheral ready or not ready; switch if off or on; insert and unload spare cassette in cass- ette drive. If this fails, take memory dump, report fault, and warm- start. Note: never load or unload disk media, or cassette media which are in use, until system has returned to to initial state.
	MCP software error	Take memory dump, report fault, and warmstart.
System initializes (PK1 and PK2 lit)	Corrupted MCP code file.	Take memory dump, report error, rerun on backup disk.
	Unexpected hardware error.	Take memory dump, report error, re- warmstart.
	Unknown MCP error.	Take memory dump, report error, attempt to rerun. If still not successful, request technical assistance.
PK lights 17 to 24 flashing (pattern other than above)	MCP has detected an error condit- ion from which it cannot recover.	Refer to section on MCP diagnostic messages for further details.
All keyboard lights flash	Memory parity	Initialize system, and rerun. If still faulty, call field engineer.



MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
Program load taking long time : keyboard disabled and D lights 4 to 7 flashing.	Program loaded thrashing due to lack of available memory; too many or too large programs in mix.	Try keyboard input. If this fails, initialize system and retry.
Hissing sound from disk drive.	Disk drive heads are damaged.	Stop work. Remove the disk from the damaged drive. THIS DISK IS PROBABLY DAMAGED, DO NOT LOAD ANY OTHER DISK INTO THE DRIVE. Call field engineer.
Non-functional disk drive.	Hardware fault on drive.	Load a good disk on drive (EXCEPT IF DRIVE HEADS HAVE CRASHED), run LS utility. If this produces errors, the drive is faulty: call field engineer.
I/O errors from disk.	Non-function disk drive.	See above.
	Crashed disk head.	See above.
	Physical damaged disk media.	Replace with new disk media.
	Disk media has bad areas.	Run CHECK.DISK to report on parity errors. Run KA to give disk directory analysis: If this shows AREA MISSING or AREA ASSIGNED TWICE, send the KA plus recent SPD message log to field engineer. Run DA to give further analysis of disk.
		Use backup disk for system files. Recover any other files by attempting to COPY them one-by-one from the corrupt disk to a good disk.

## **B 90 DEPENDENT UTILITIES**

The following pages describe those system utilities which run under MCP control but which are relevant only to the B 90 CMS software.

## CONFIGURER (Configure B 90 Software System)

This utility sets up data in a disk file called "SYSCONFIG" to determine how the B 90 is to be used. The SYSCONFIG file must be present on the system disk if any of the software options are to be used, other than the defaults given later. The data in this file is examined at warmstart time. Therefore any options specified will only be effective after the next warmstart.

To execute, enter  
CONFIGURER

The utility runs in an interactive mode, using the console as input.

"SPO LOGGING REQUIRED Y OR N". Enter "Y" if a log of SPO messages is to be kept on disk. This will normally be required if the SPO messages are sent to the self-scan display where there is no hard-copy. If no SPO logging is required, enter "N".

"ENTER NUMBER OF LOG FILES". Enter a number between 3 and 16. At warmstart this number of files will be created on the system disk to hold the logs. The names of the files are "SYS-LOG-01" to "SYS-LOG-*nn*" where "*n*" is the number entered. The minimum is 3 files.

"ENTER LOG FILE SIZE IN SECTORS". Enter a number which will be the size of each of the log files, between 32 and 16383. The minimum log file size is 32 sectors. Therefore the minimum disk space for log files is (32 x 3) sectors.

"INFORMATION FOR WARM START: ENTER ID OF FILE TO BE ZIPPED". If it is not desired to execute a program at the beginning of each warmstart, enter the name of the program code file (including disk-id if not on the system disk) to be executed. If no program is to be run, make a null input (just OCK1).

"ENTER STARTUP MESSAGE". Enter up to 80 characters, terminated by OCK1, as a warmstart message. This message will be displayed at the beginning of each warmstart. If no message is required, just depress OCK1.

"ENTER POWER OFF MESSAGE". Enter up to 80 characters, terminated by OCK1, as a message to be displayed when the system disk is powered off. If no message is required, just depress OCK1.

"SPO OPTION SSA OR SPA". Enter "SSA" if it is desired to display SPO messages on the self-scan display. This will free the console printer to be used entirely by programs with console files. The self-scan can also be used by programs. When the self-scan is used both programmatically and as a SPO, the SPO messages are displayed when the ready enable key and OCK1 are depressed.

Enter "SPA" if it is desired to write SPO messages to the serial printer. This is especially useful if the console printer is equipped with more than one tractor, in which case only the bottom left-hand tractor will be used by the MCP for SPO messages, and other tractors can be used exclusively by programs with console files.

"JOB COMPLETED". This is displayed when the utility terminates.

### Defaults:

If the SYSCONFIG file is not present on the system disk, the following options are assumed:

SPO logging is enabled.

The number of log files is 3.

The size of each log file is 32 sectors.

No message is displayed at startup.

The SPO option is SPA.

Output Messages:

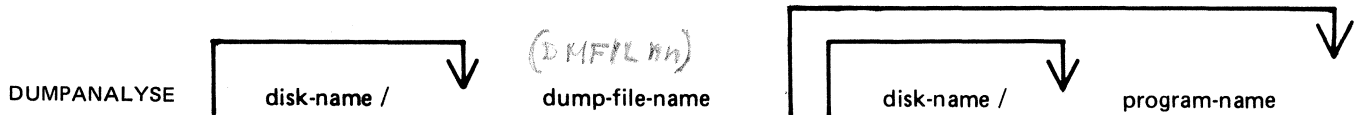
MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
ILLEGAL NUMBER OF LOG FILES	A number outside the range 3-16 was entered.	Wait for repeat, then enter correct value.
ILLEGAL FILE SIZE	A number outside the range 32-16383 was entered.	wait for repeat, prompt, then enter correct value.
ILLEGAL -	A file-name entered does not have the correct format.	Wait for repeat prompt, then enter correct value.
INVALID CHARACTER IN IDENTIFIER identifier	Typing mistake	Check input and re-enter when prompted.
**INVALID SELECTION - RETRY**	Option has been specified which is not one of allowed values.	Wait for repeat prompt, then enter correct information.
PARITY ERROR ON WRITE TO FILE file-name. ERROR AT RECORD NUMBER number.	Parity error occurred while writing information to the SYSCONFIG file. Utility terminates.	Investigate cause of disk error. Use different disk. Repeat execution of program.

## DUMPANALYSE (Analyze B 90 Program Dump Files)

When a program is DP'ed (see DP intrinsic) a file on the system disk is created with a name DMFILnn where nn is the mix number of the program dumped. This utility provides an analysis of this file, for use by technical analysts, printed on the line printer.

If the dumped program was written in MPL (used BILINTERPX) only the dump-file is required for a full analysis. If the dumped program was written in COBOL or RPG (used COBOLINTX), a fuller analysis can be provided by specifying the program code file as well.

Format:



Note that the program name must be the name specified at compile time and placed in the code file. This is normally the same as the disk file name but the file name could have been changed by the CH utility. If the wrong program name (or none at all) is given for a COBOL or RPG program, then the dump analysis will be incomplete (no COP table data can be analyzed).

Examples:

To analyze the dump file of an MPL program DP'ed when it was mix number 1:

```
DUMPANALYSE DMFIL01
```

To analyze a dump file (which had been copied to a new file DPFIL on disk PRB) caused by DP'ing the COBOL program AR678:

```
DUMPANALYSE PRB/DPFILE AR678
```

To analyze the dump file created by DP'ing the RPG program RS202P. The program was at mix number 3 and the code file resides on disk RTB:

```
DUMPANALYSE DMFIL03 RTB/RS202P
```

On completion of the printed analysis, the dump file is removed from disk. To preserve a copy of the dump file, make a backup copy (using the COPY program) before executing the DUMPANALYSE program.

Output Messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
**WARNING** BIL DUMP FL, SECOND PARAMETER IN INIT.MESS IGNORED	Program name has been specified for a dump file from an MPL program.	None.
PACK ID TOO LONG disk-name	Disk-name exceeds 7 characters. Util- ity terminates.	Check input and re-enter.

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
FILE ID TOO LONG file-name	File-name exceeds 12 characters. Utility terminates.	Check input and re-enter.
INVALID CHARACTERS IN FILE file-name	File-name or program name mis-spelled. Utility terminates.	Check input and re-enter.
file-name NOT FOUND	Dump file or program file cannot be found. Utility terminates.	Check input and re-enter. Or make file present by loading program or dump file from backup copy.
INTERPRETER VERSION NOT SUPPORTED interpreter-name	Dump file is corr- upted; Error in com- piler creating prog- ram file.	Use backup copy of dump file; retain program code file and request tech- nical assistance.

## GEN.DUMPFL (Create Empty B 90 Memory Dump File)

Before the contents of memory can be dumped to disk, an empty disk file called MEMDUMP must be created. This file must be large enough to take the contents of all the memory of the system on which the dump is being taken.

The size of the file is specified by "number" in the range 64 to 11264. If "number" is not given, the size defaults to actual memory size of the machine in use.

Format:

GEN.DUMPFL    number    ON disk-name    AS file-name

Examples:

To create an empty file called MEMDUMP on the system disk for a B 90 with 80K bytes of memory:

GEN.DUMPFL 80

To create an empty file called MEMORY.DUMP on disk PRB for a 128K B 90:

GEN.DUMPFL 128 ON PRB AS MEMORY.DUMP

Note that this file-name must be changed to "MEMDUMP" before a successful memory dump can be taken. Any existing file of the same name will be removed when the new file is created by this utility.

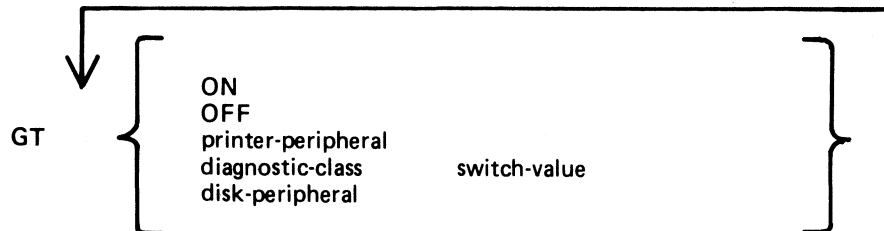
Output Messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
SPACE RESERVED	Successful EOJ	None.
SIZE TOO SMALL	"number" is less than 64.	Re-input.
SIZE TOO LARGE	"number" is greater than 11264.	Re-input.
DISK NAME TOO LARGE	Disk name is greater than 7 characters.	Re-input.
FILE NAME TOO LONG	File name is greater than 12 characters.	Re-input.

## GT (General Trace)

The general trace command is an MCP intrinsic which displays various diagnostic information either on the system console or on the line printer. The general format is:

Format:



To turn the trace printing on, with printing on the console (SPA) enter

```
GT ON
```

To direct the diagnostic printing to a line printer, specify the printer peripheral. For example

```
GT LPA
```

Note that the trace must also be turned ON in this case. The trace will be interleaved with any program printout.

To turn the trace off, enter

```
GT OFF
```

To display the cumulative number of retries performed on a disk unit since the last warmstart, specify the desired peripheral. For example,

```
GT DMA
```

The general B 90 machine code trace is implemented as follows:

Each trace point is identified by a diagnostic class and a diagnostic value. The class is one of 16, identified by a hex digit (0-F). This identifies the system function being performed, as follows:

0	Open/close file handling routines
1	Indexed file handling
2	Accept/display routines
3-7	Intrinsic functions (for example, SORTINTRINS)
8	Automatic volume recognition (AVR) routines
9	BAILIFF (task handling MCP routine)
A	Disk space allocate/deallocate routines
B	Interpreters (BILINTERPX, COBOLINTX, NDL.INTERPX)
C	MCP communicate handler (MCH)
D	Virtual memory (VM) routines
E	Task control routines (for example, BAILIFF)
F	I/O master interrupt handler (MIP)



The diagnostic value is a hex number (0-F) giving a measure of depth of trace required (0 is least significant, F is critical).

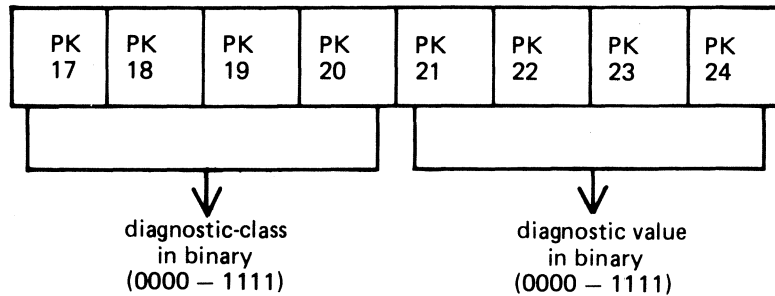
The GT command allows the storage of a switch-value for each diagnostic class. If the trace has been initiated via the GT ON command, then each time a trace point is encountered in the machine code, a diagnostic printout will occur if the switch-value for that class of trace is lower than or equal to the diagnostic value of that trace point. For example, if a particular trace point had a diagnostic value of C, then a trace point would occur if the switch-value for the appropriate class was in the range 0-C inclusive. If the switch-value set by the GT command was D, E or F, no trace print would occur.

The default values set at warmstart are:

- trace option OFF
- diagnostic print directed to SPA
- all switch-values set to F.

Therefore the only trace prints that will normally occur are the critical ones (diagnostic value F).

Note that if the trace option is ON the PK lights display the current trace point, whether or not the printing takes place, as follows:



For example, a trace point with class D (binary 1101) and severity 7 (binary 0111) will cause PK lights 17, 18, 20, 22, 23 and 24 to be lit.

The switch-value entered in the GT command must contain two hex digits:

- first digit - switch-value for register diagnostics
- second digit - switch-value for memory diagnostics

If no options are specified for the GT command, then the current switch-values for each of the 16 diagnostic classes is displayed.

Examples:

To find the retry count for DMA and DKA:

```

GT DMA
DMA      0      RETRIES

GT DKA
GT DKA NOT ON SYSTEM

```

To set the switch-values to CC for interpreters, then interrogate all switch-values, then turn on printing to LPA:

```

GT      B      CC
GT
          DIAG SWITCHES FF FF FF FF FF FF FF FF FF
.....FF FF CC FF FF FF FF
GT LPA
GT ON

```

Note that only the disk-peripheral option and the GT command with no further options result in an immediate response to the operator.

Format of diagnostic printout:

The format of a register diagnostic message is given here, where each X represents a single hexadecimal digit. The hex string is printed on one line.

register :	AD	BO	B1FL	J	K	L
	xx	xx	xxxx	xxxx	xxxx	xxxx

register	M1	M2	WR	X	Y	MXA	MXB	UMRX	AD,ESCT
	xxxx	xxxx	xxxx	xxxxxxxx	xxxxxxxx	xxxx	xxxx	xxxx	xxxx

The diagnostic class and value are given by the AD register (first 2 digits, repeated in first pair of last four digits). For example, a DF diagnostic with M1 = 1111 would look like:

```

DF xx xxxx xxxx xxxx xxxx
          1111 xxxx xxxx xxxxxxxx xxxxxxxx xxxx xxxx xxxx DF xx

```

where x indicates any hexadecimal digit.

When GT is switched on, system fatal errors (which would normally result in a set of PK lights 17 to 24 flashing) are reported as a diagnostic printed message followed by initialization to the Initial state. A memory dump should normally be taken to find more information. When GT is switched off, the pattern of PK lights 17 to 24 that are set flashing on a system fatal error correspond to the value of the AD register.

## ND (New Density)

This intrinsic allows the operator to define the print density on suitable console printers.

Format:

ND peripheral density
-----------------------

The density field is a single character which specifies the density as follows:

A. The greatest number of characters per inch available on the printer.

B. The second highest number of characters per inch.

and so on.

1. The greatest number of lines per inch available on the printer.

2. The second highest number of lines per inch.

and so on.

Examples:

ND SPA 1

ND SPA B

When the number of characters per inch changes, the operating system will adjust the values of page width and offset (previously set by an FD command or by default) so that any subsequent output is restricted to the part of the platten available with the previous density. The page height will be adjusted when the number of lines per inch changes.

The new density, together with the adjusted values of page height or page width and offset, are recorded in the system configuration (SYSCONFIG) file and are remembered across system shutdown and warmstart.

At warmstart, if any inconsistency is found between the system configuration file information and the capability of the current printer, then the system configuration file information is ignored and default values are used. The system configuration file is unaltered.

The densities available on different printers are as follows:

PERIPHERAL	DENSITY	VALUE
120 cps console printer	15 characters per inch	A
	10 characters per inch	B
90 cps console printer	15 characters per inch	A
	10 characters per inch	B
	8 lines per inch	1
	6 lines per inch	2

Output Messages:

MESSAGE	POSSIBLE CAUSE	SUGGESTED ACTION
input INVALID	(1)the specified density value exceeds the number of options available on the configured console (2)the printer is in use (3)the mnemonic is incorrect	(1)enter correct input  (2)wait until program has closed the console file correct the input and re-enter
input NOT ON SYSTEM	The specified peripheral does not exist	Correct the input and re-enter

## PATCHMAKER (Patch B 90 Machine-Code Object Program Files)

This utility reads a file of patches from disk or cassette or from console keyboard input and patches a machine-coded system software item. Stringent conditions are enforced to make the patch, including the necessity to apply each patch in the correct order. All previous patches must be applied before making the next patch.

Note: it is essential that an unused copy of all micro-coded software items is retained for patching. It is not possible to patch a B 90 MCP that has been used in normal B 90 operation. This is because certain MCP tables included in the code file are modified during operation. This modification would cause the check digit calculations in PATCHMAKER to fail. The Stand-Alone Utility (SAU) COPY function may be used to create unused copies of all system software for patching purposes, and also to create backup copies of patched software. Files on the system disk cannot be patched: they must reside on a user disk.

To execute, enter

PATCHMAKER

The utility runs in an interactive mode.

The utility displays on the SPO: "IS PATCH FILE TO BE ENTERED FORM CONSOLE"

and waits on an ACCEPT.

If the operator enters

AX mix-number YES

then the console file will be opened. Any other response causes the utility to ask, via displays on the SPO, if the file containing patches is on a cassette: if the response is "YES" a tape file named "PATCHES" is required, and any other response requires a disk file on the system disk named "PATCHES".

If a console file is opened the utility displays

{ " ?DATA PATCHES  
SIGNIFY WHETHER PATCH FILE IS TO BE  
OUTPUT ON CASSETTE OR DISK"

The operator may enter, via an ACCEPT, either "CASSETTE" or "DISK". Patches entered subsequently on the keyboard will be written to the specified medium.

The utility displays

"ENTER PATCHES NOW"

The patches must be entered via the keyboard from the hard-copy provided. The characters and terminating keys must be entered exactly as supplied, although spaces are not significant and may be entered as found convenient. The utility will ask for resubmission of lines which are obviously incorrect.

If correct, the message

"PATCHES HAVE BEEN ENTERED CORRECTLY

"?END PATCHES"

is given.

When the correct patch has been entered, the utility displays on the SPO

"IS PATCHING NOW REQUIRED"

and waits on an ACCEPT.

If the operator enters anything other than  
AX mix-number YES

then the utility will go to normal EOJ. If the operator enters "YES", then patching will be carried out.

The utility displays

"ENTER DISK IDENTITY OF FILES TO BE PATCHED".

and waits on an ACCEPT.

The operator must enter the disk name of the disk on which reside all the software files to be patched. This must not be the system disk.

The utility displays

"FILE file-name BEING PATCHED,  
TO BE SAVED AS "

and waits on an ACCEPT.

The file-name is the name of the file to be patched, which is generated from the patch itself. This file must be present on the disk specified earlier. The operator must enter the new name for the file after it has been patched. The patched file will be retained only if the patching is successful.

Checksums are computed and verified before and after patching. If the utility goes to EOJ without displaying any error messages then the patching has been successful.

#### Output Messages:

MESSAGE	POSSIBLE CAUSES	SUGGESTED ACTION
ERROR IN PATCHES ENTERED - RUN ABORTED	Keyboard input has been made in error.	Check keyboard input with hard-copy, re- execute utility.
PATCHES TO file- name-1 SUCCESSFULLY ACCOMPLISHD AND SAVED IN file- name-2	Successful patch- ing run.	Make backup copy of file-name-2 (unused patched file) for future patching.
INITIAL CHECKSUM DISCREPANCY	Wrong software file has been submitted for patching.	Re-execute utility with correct input file.
Other error mess- ages	Wrong input file; other serious soft- ware errors.	Request technical assistance.

Note: the device kind of the utility's console file may be modified via the MODIFY utility to one of KB, KD or KP if required. The internal name of the console file is THREADS.

## PMB90 (Analyze B 90 Memory Dumps)

This utility is an interactive program which produces a formatted print of the contents of a memory dump tape or disk file produced by the memory dump feature of the B 90 bootstrap ROM. The tape must be labelled "MEMDUMP/MEMORY" or the disk file must be named MEMDUMP on the system disk unless otherwise specified to PMB90.

The utility requires the following files on the system disk:

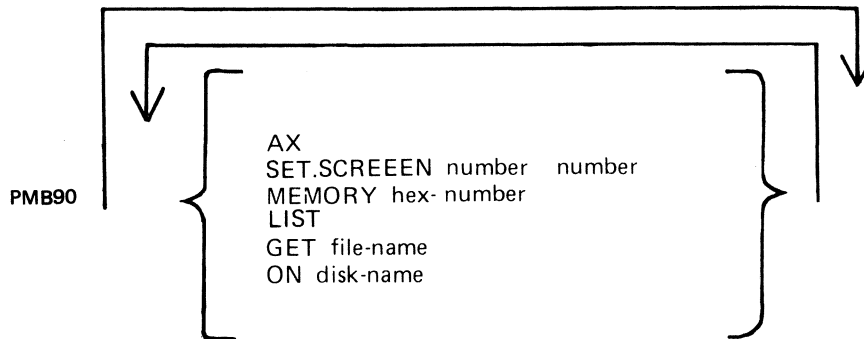
- PMB90 – object code file
- PMBHELP – data file of prompt messages
- PMBERROR – data file of error messages
- PMBM.xxxxx – data file for information on MCP xxxxx
- PMBO.xxxxx – data file for more information on MCP xxxxx

The x values vary with each release: the files provided with each release must be used with that release, otherwise incorrect analysis may be made.

### Starting the Utility

The utility can be executed with a number of options in the initial message. The format is as follows:

Format:



The meaning of these options is as follows:

#### AX

The program will use the SPO via DISPLAYS and ACCEPTS to communicate with the operator. If this option is not specified the console will be used for communication.

#### SET.SCREEN

This option sets the screen page and line sizes from the numbers specified. These must be set if the DISPLAY option is used (see later). The input medium is the console keyboard but echoing of input is on the self-scan screen.

#### MEMORY → NOW USE ↓

If this option is specified, the program will read only the first part of the memory dump, up to the byte given by the hexadecimal number. Note that use of this option could cause the program to fail on certain print options if the analysis requires a part of the dumped memory that has been excluded.





## SAVE

This option enables a copy of the dump (patched if required) to be made on the specified disk with the specified file-name.

## PATCH

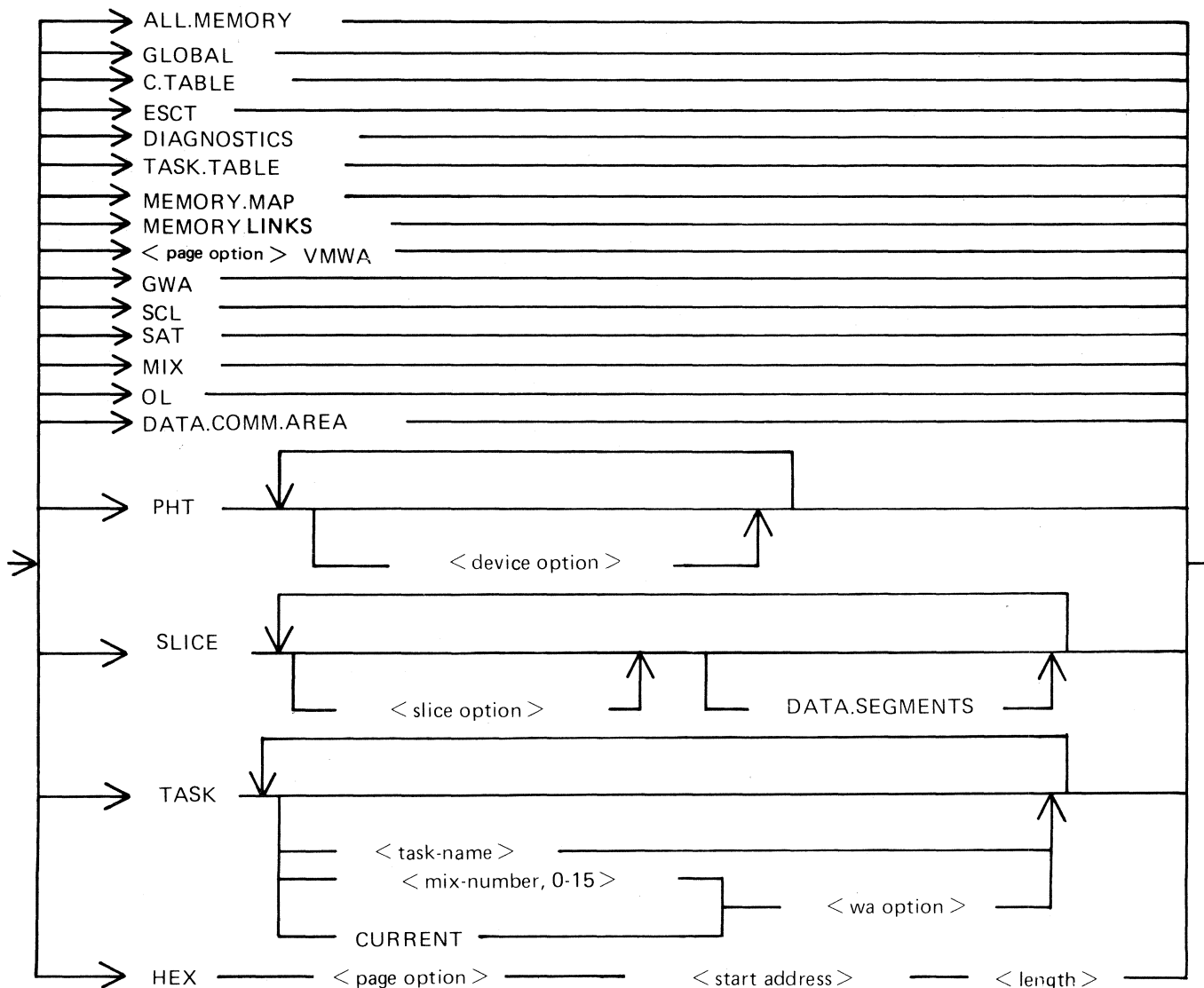
This option enables invalid areas of memory to be patched in the dump, to enable PMB80 to continue its analysis.

The <page-option> may only be ONE. The <address> is four hex digits, and the <hex-addr> may be from 1 to 16 hex digits.

## END, BYE

These alternative options cause PMB80 to go to normal EOJ.

The format of <print-option> is as follows:



## ALL.MEMORY

This option creates an analysis of fixed MCP data areas, configuration table and task table if present, all peripheral handler tables, memory links and analysis of all locked slices in memory plus all overlayable code and data segments present in memory at the time of the dump.

## GLOBAL

This option creates an analysis of the fixed MCP data areas only

## C.TABLE

This option gives a print of the configuration table if it was in memory at the time when the dump was taken.

## ESCT

This option gives a print of the mix table (execution scan priority table).

## DIAGNOSTICS

This option gives a print of the MCP's diagnostic buffer area.

## TASK.TABLE

This option gives further analysis of the mix, if the overlayable task table was present in memory at the time when the dump was taken.

## MEMORY.MAP

This option provides an analysis of the layout of memory.

## MEMORY.LINKS

This option analyzes the layout of the overlayable area of memory.

## VMWA

This option gives a print of the virtual memory work area only.

## GWA

This option gives a print of the Global work area only.

## SCL

This option prints the keyboard buffer only.

## SAT

This option gives a print of the Slice Address Table only.

## MIX

This option gives a selective analysis of parts of the dump relating to the tasks running at the time of the dump.

## OL

This option provides a selective analysis of peripheral configuration information.

## DATA.COMM.AREA

This option prints the areas of memory relating to data communications.

## PHT

This option gives a print of selected Peripheral Handler Tables. If the <device-option>s are absent, then all peripherals attached to the system at the time of the memory dump are analyzed. Allowable values for <device-option> are:

- CX – Channel Expander
- LP – Line Printers
- SP – Serial (console) Printer
- CT – Cassettes
- DK – Cartridge Disks
- DF – Fixed Disks
- DM – BSM Disks
- KB – Keyboard
- SS – Self-Scan
- ADC – Asynchronous Data Comm Controllers
- SDC – Synchronous Data Comm Controllers
- DI – ICMDs
- SDI – BSMII Disks

## SLICE

This option provides selective printing of locked slices of memory or of data segments. The <slice-option> may be either a “slice number” in the range 0-45 or one of the following names:

- DISKDDR
- LPDDR
- PANDDR
- KBDDR
- CASSDDR
- SENDDDR
- CONSOLE
- INXS
- SCREENSN
- SUSN
- INITIALIZE
- ADCDDR
- SDCDDR
- OPENCLOSE
- DCCH
- SPO
- CONBUFSN
- SCLBUFSN
- ICMDDDR
- OCOMSN
- DIAGSN

## TASK

This option prints the contents of a Task Control Block (TCB).

The <task-name> may be one of

- NDL
- MCS
- BAILIFF
- SCL
- LOADER

The <wa-option> may be one of

- MPLII
- BIL
- COBOL
- RPG
- SORT
- NDL

## HEX

This option provides a print (or display) in hexadecimal and byte format of selected parts of memory. The <start address> is a four-hex-digit number and the <length> is also specified as a four-hex digit number.

### Example:

To obtain a complete memory dump print on the console printer:

```
PMB90 (OCK)
PRINT ALL.MEMORY (OCK)
```

To obtain a dump of the data comm buffers, plus the data comm controller device-dependent routines, plus the MCS and NDL task tables:

```
PMB90 (OCK)
PRINT DATA.COMM.AREA (OCK)
PRINT PHT ADC SDC (OCK)
PRINT TASK MCS NDL (OCK)
```

### Note:

When submitting memory dumps for analysis, it is helpful if some preliminary analysis has already been performed. The following option is recommended:

```
PMB90
PRINT MIX OL MEMORY.MAP MEMORY.LINKS GLOBAL PHT TASK CURRENT
```

Always provide the MEMDUMP file on magnetic media even if this preliminary analysis has been performed.

## POWER OFF

Logically power off all user disks (see PO command if under MCP control, or SAU PO command if under SAU control).

Logically power off the system disk (see PO command if under MCP control). Wait until the system returns to the initial state, that is PK1 and PK2 are lit.

If the PO command cannot be used, due to some system error, then the system should be halted by pressing the Load Enable button, causing the system to return to the initial state with PK1 and PK2 lit.

Remove all removable disk media.

A mini disk can be removed immediately the unit door is opened.

A disk cartridge can be removed only when the red stop light is lit, assuming that the drive is functioning correctly.

Power off the disk units (failure to remove disk media before this, may result in subsequent media corruption).

Remove all cassettes from the system.

Power off the main cabinet (this must be the LAST action after all peripherals have been switched off).

Note on disk removal:

There are only two situations when it is valid to remove a disk:

where the MCP is not running and the disk is not in use.

where the MCP is running, but the disk is a user disk which is logically powered off after using the PO command: note that the PO command does not cause a disk to become logically powered off if it is in use, but the PO will be completed only after all activity on the disk is complete.

Note on power failures:

If the main cabinet is switched off accidentally (for example, by power failure), remove all disks and cassettes before it is switched back on.

**SECTION 10  
B 1800/B 1900-DEPENDENT  
SYSTEM SOFTWARE**

**TO BE PROVIDED**

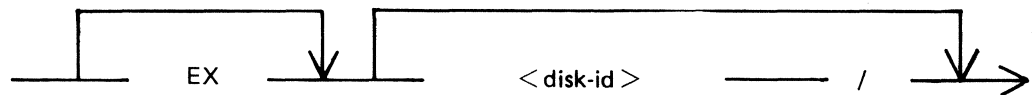
# APPENDIX A

## COMPLETE RAILROAD DIAGRAMS

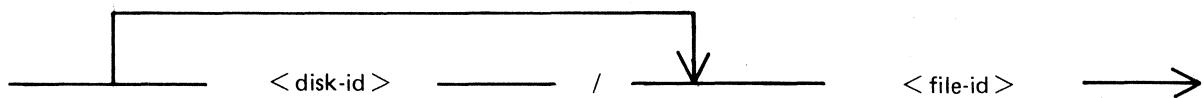
This appendix gives the railroad diagrams for all the CMS-common intrinsics and utilities, including SORT and CO, in alphabetical order. These diagrams give the complete input message formats, for ease of reference.

For details of the meaning of these messages, refer to the text.

In the following diagrams the `<ex-option>` is defined as :



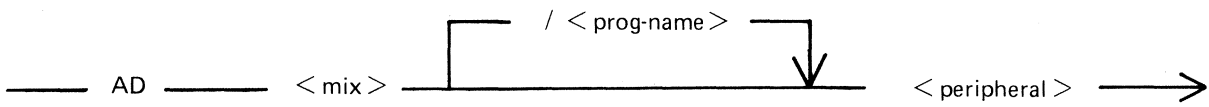
and `< file-name >` is defined as :



---

**AD intrinsic**

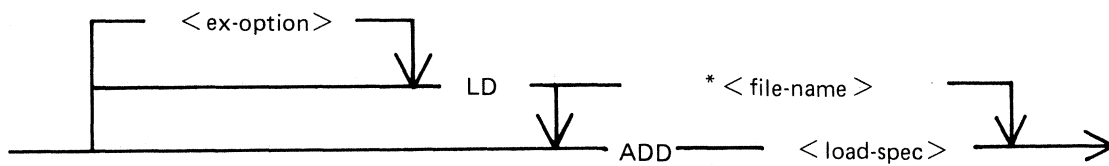
---



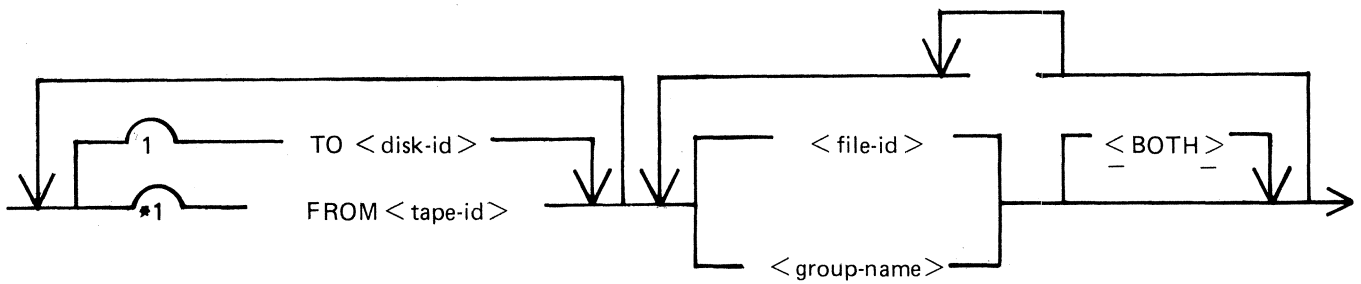
---

**ADD utility**

---



<load-spec> is defined as :

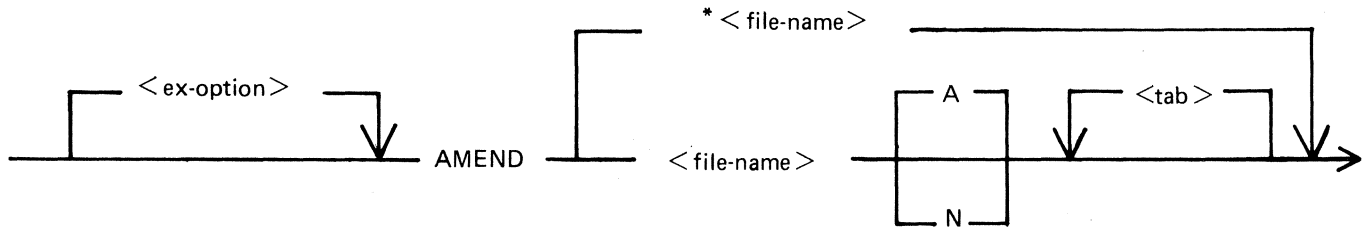




---

## AMEND utility

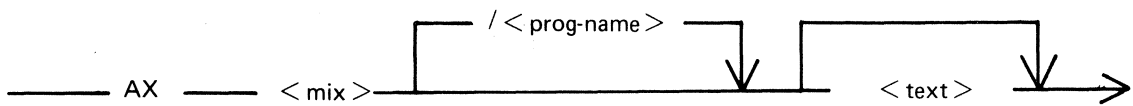
---




---

## AX intrinsic

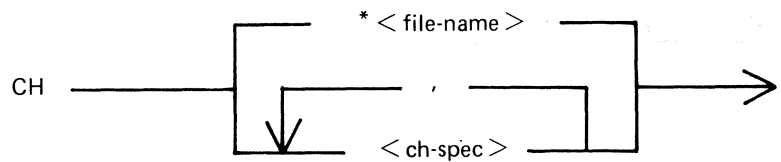
---



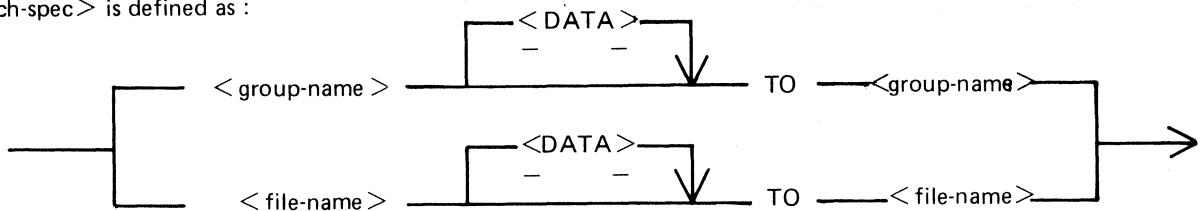

---

## CH utility

---



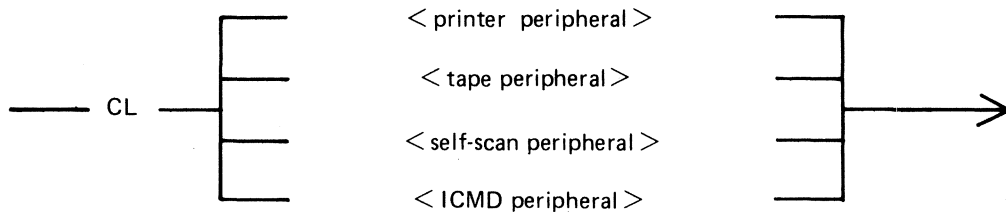
<ch-spec> is defined as :



---

**CL intrinsic**

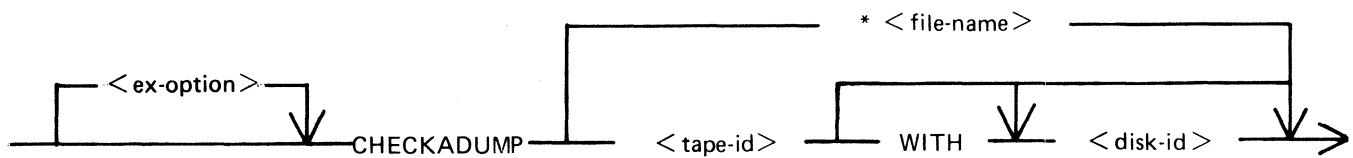
---



---

**CHECKADUMP utility**

---



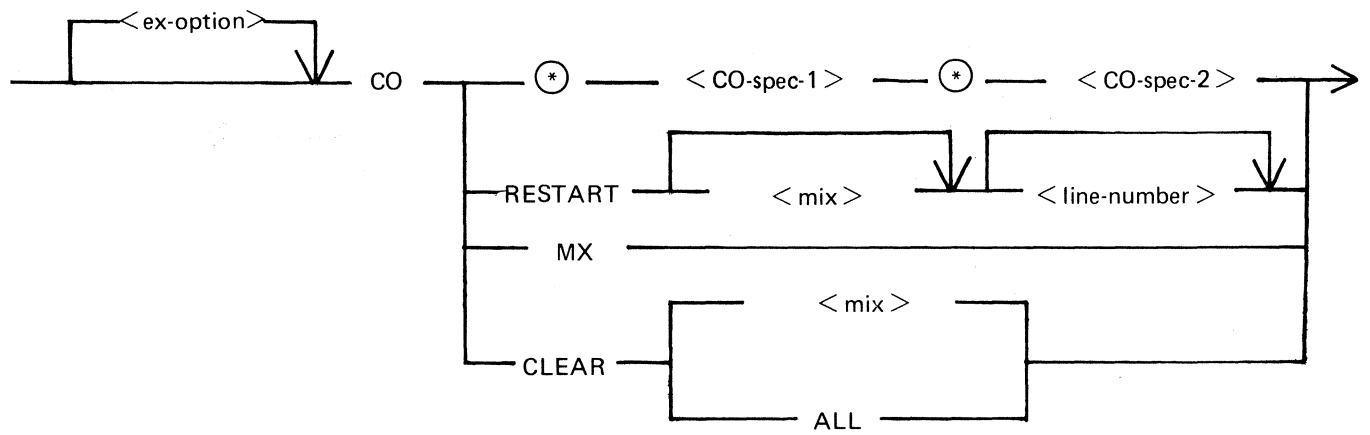
---

**CHECK.DISK utility**

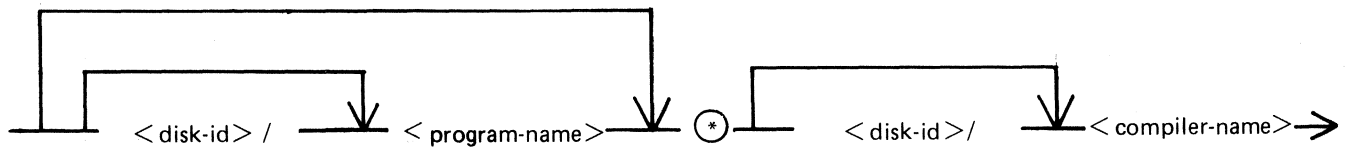
---



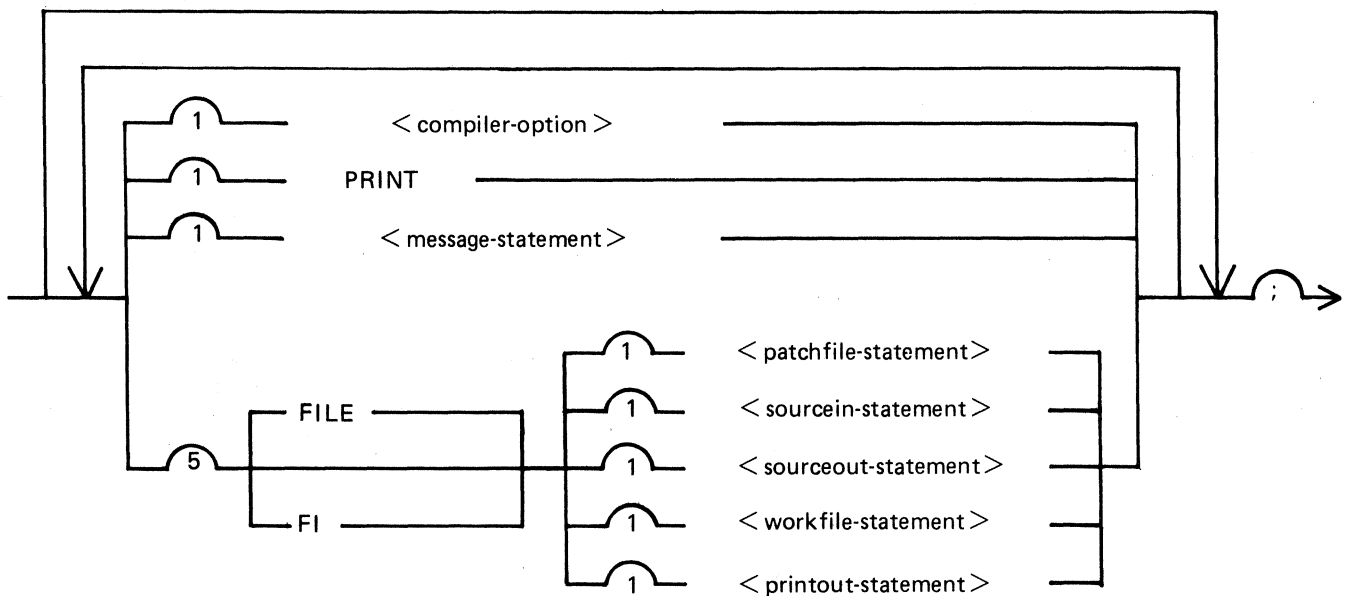
**CO utility**



<CO-spec-1> is defined as :



<CO-spec-2> is defined as :



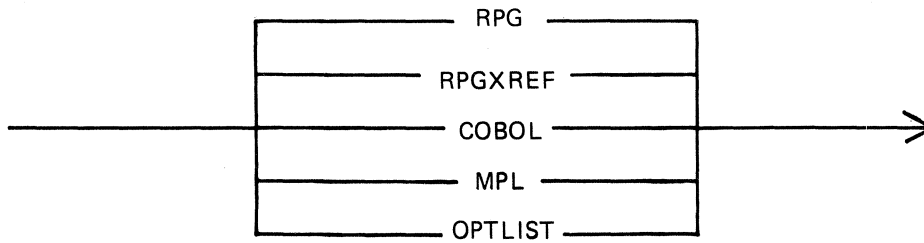
\* indicates the allowed positions for macro cells

---

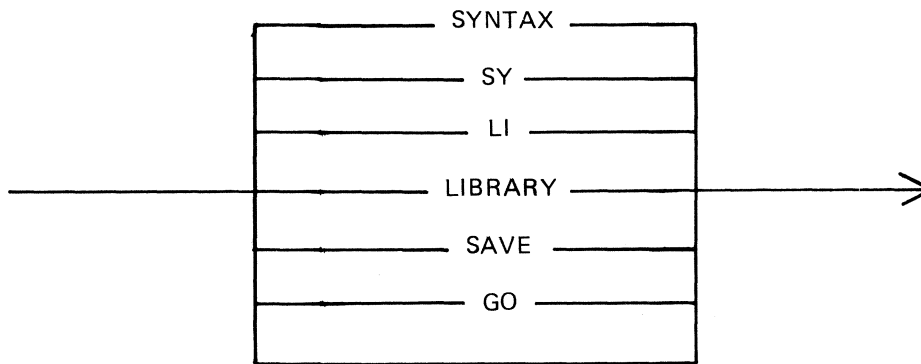
**CO utility (Continued)**

---

The < compiler name > is defined as :



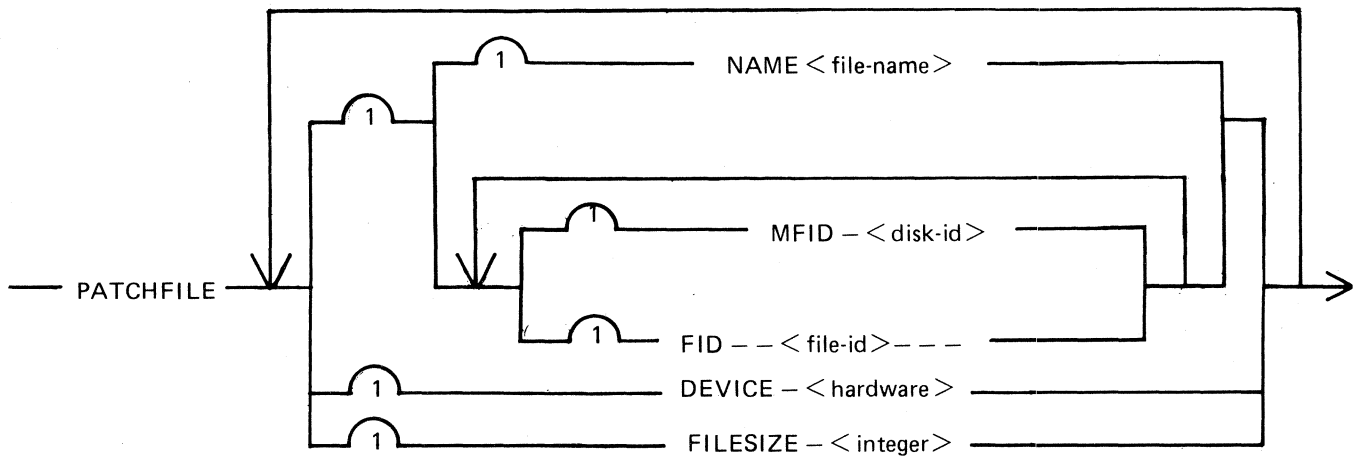
The < compiler-option > is defined as :



The < message-statement > is defined as :

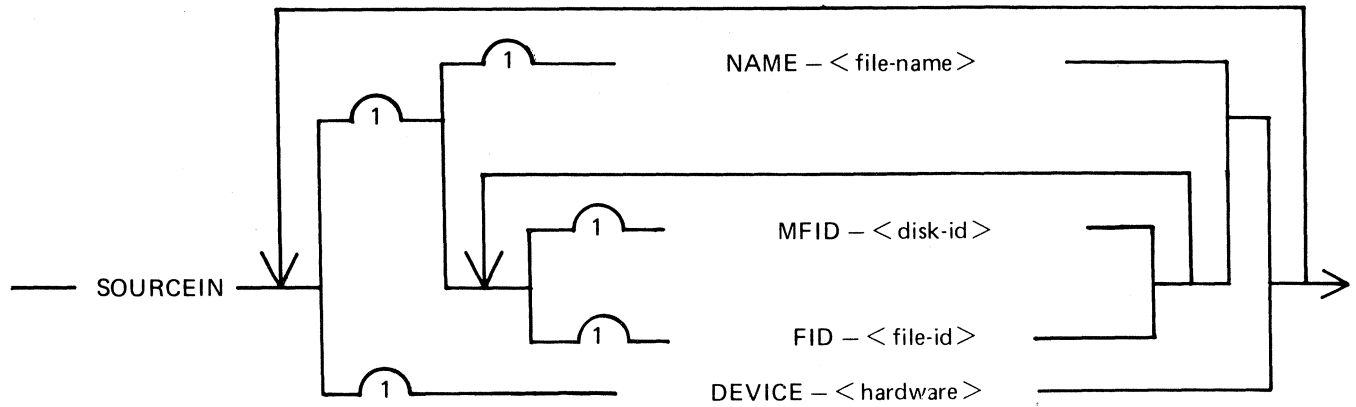


< patchfile-statement > is defined as :

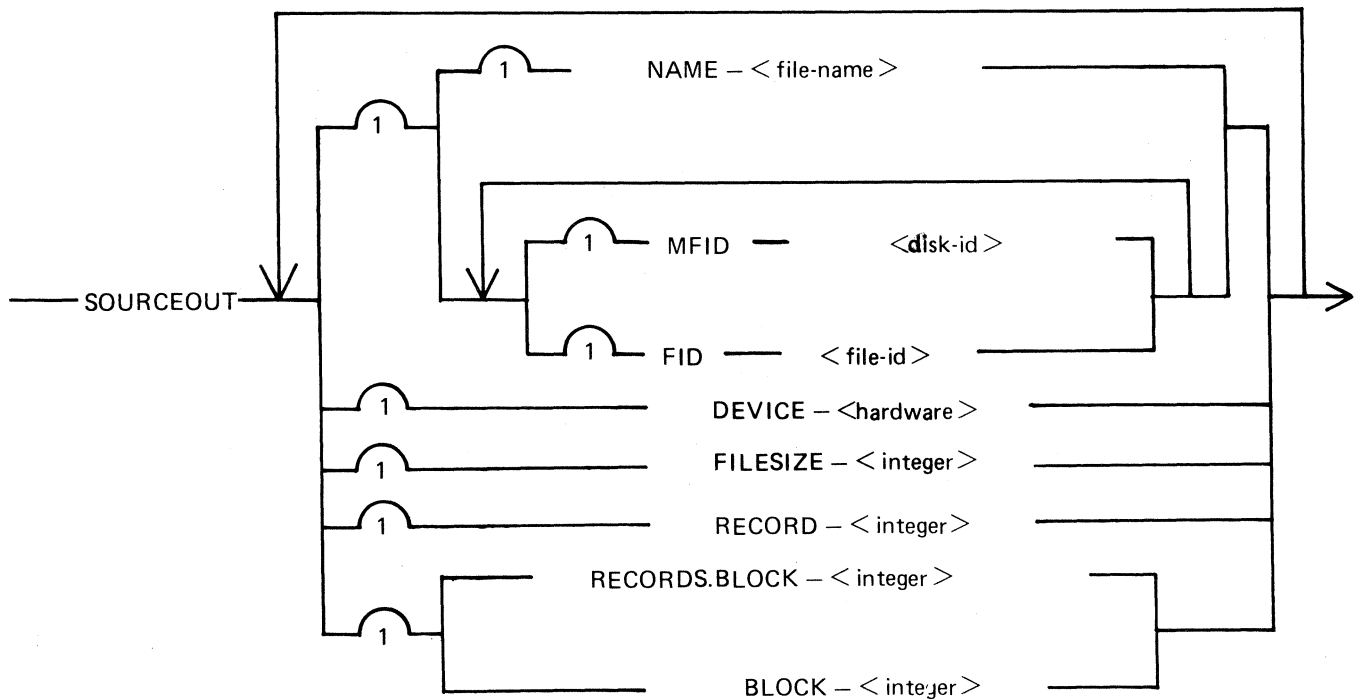


**CO utility (Continued)**

< sourcein-statement > is defined as :



< sourceout-statement > is defined as :



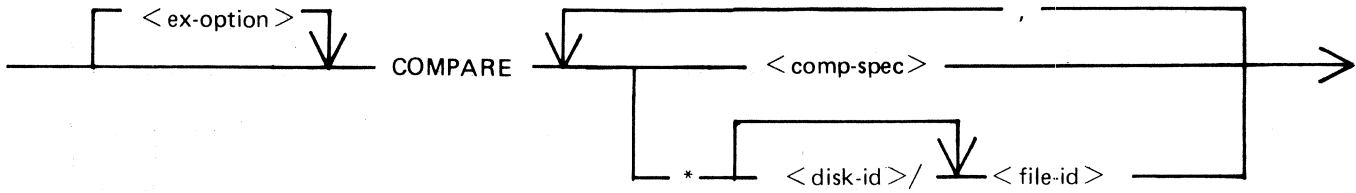
< workfile-statement > is defined as :

—WORKFILE — MFID — < disk-id > —

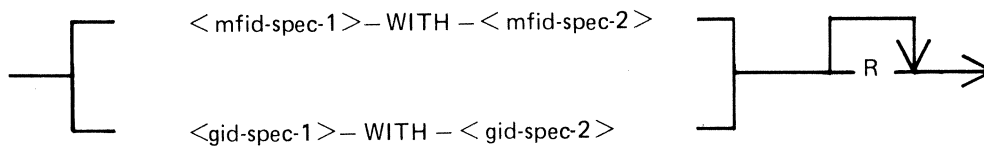
< printout-statement > is defined as :

—PRINTOUT — DEVICE — < hardware > —

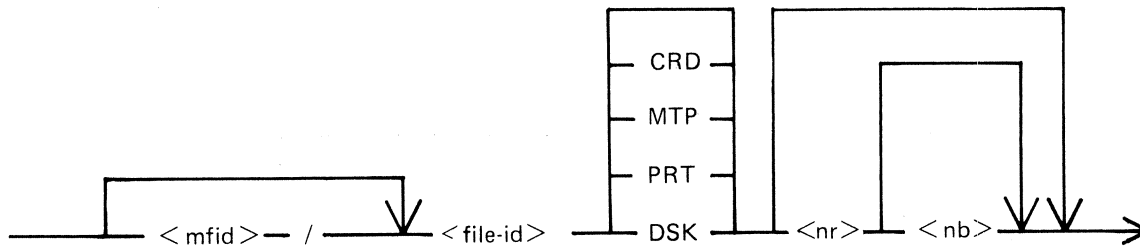
## COMPARE utility



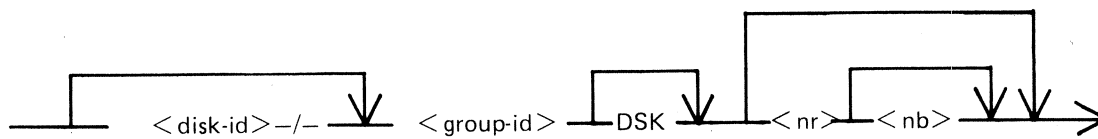
<comp-spec> is defined as :



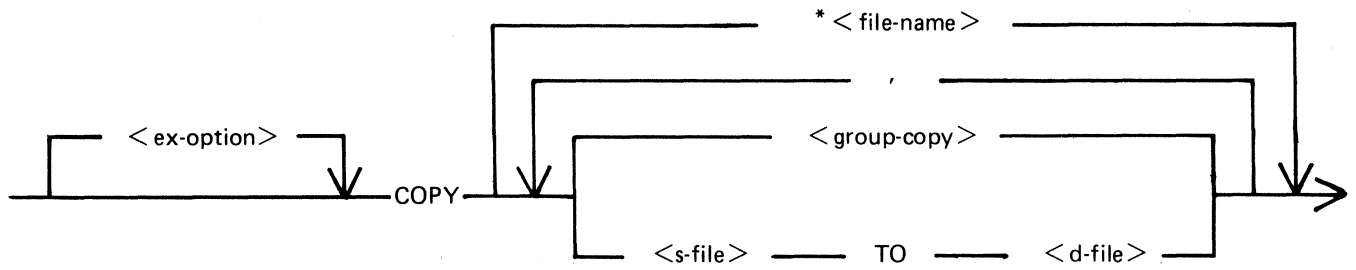
<mfid-spec> is defined as :



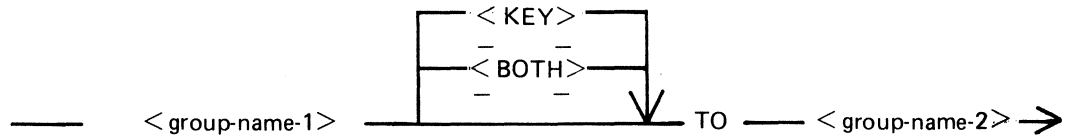
<gid-spec> is defined as :



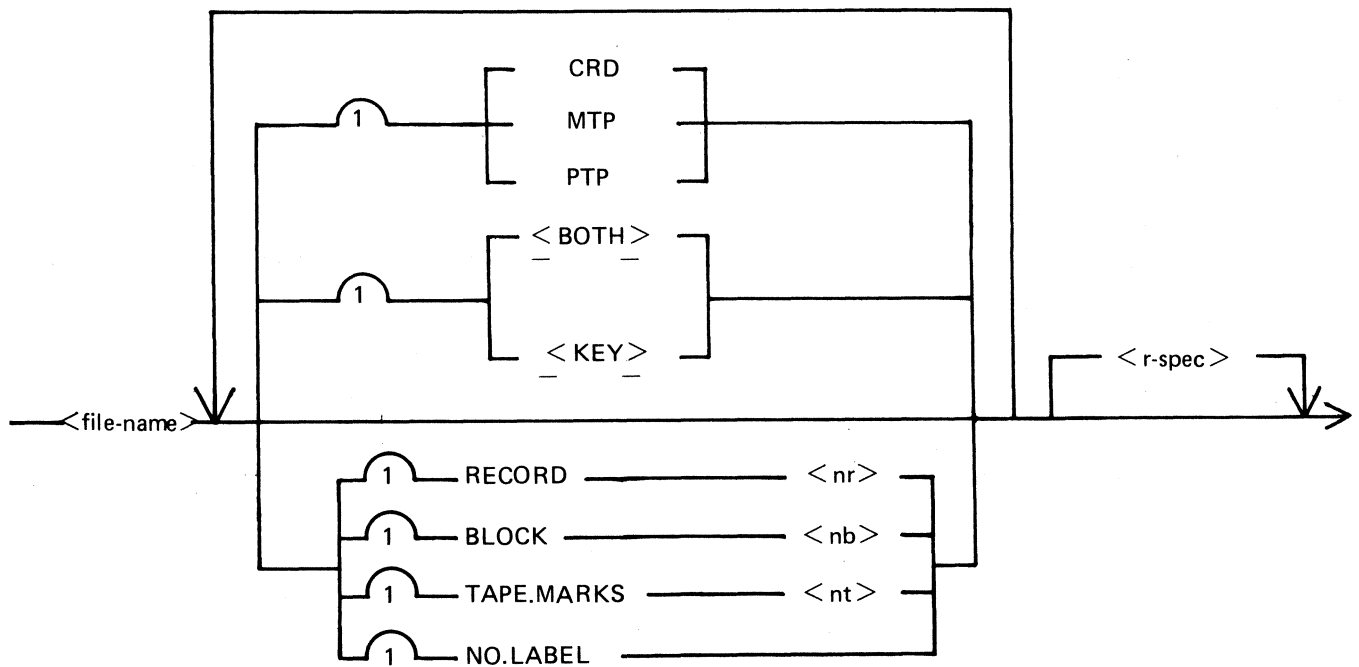
**COPY utility**



`<group-copy>` is defined as :

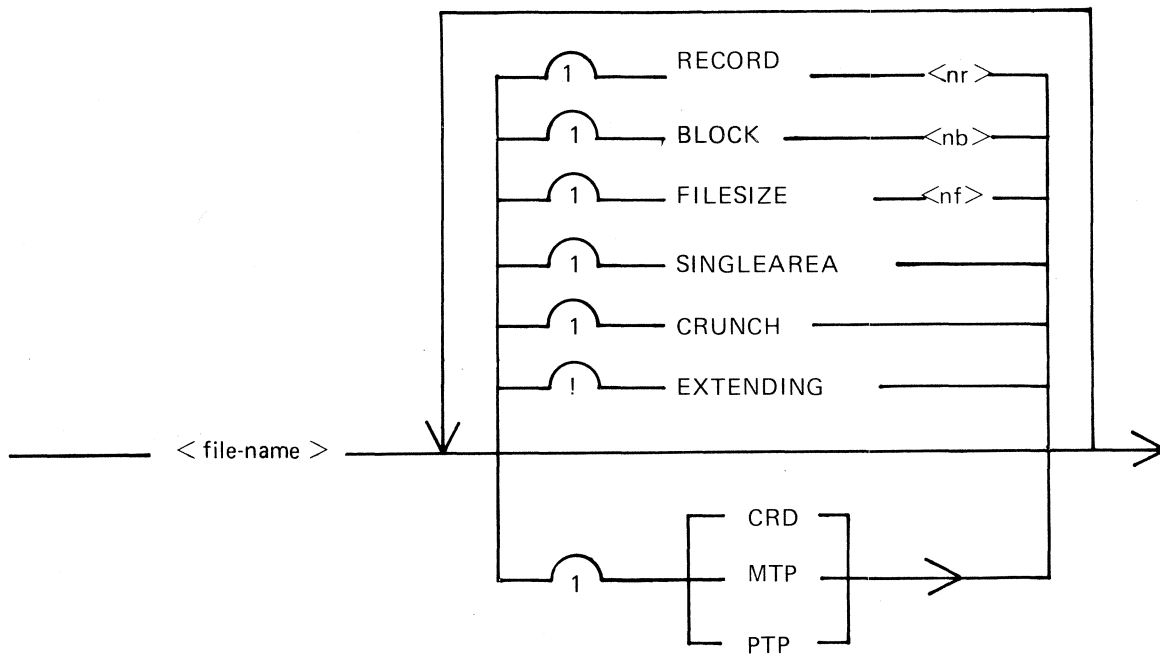


`<s-file>` is defined as :

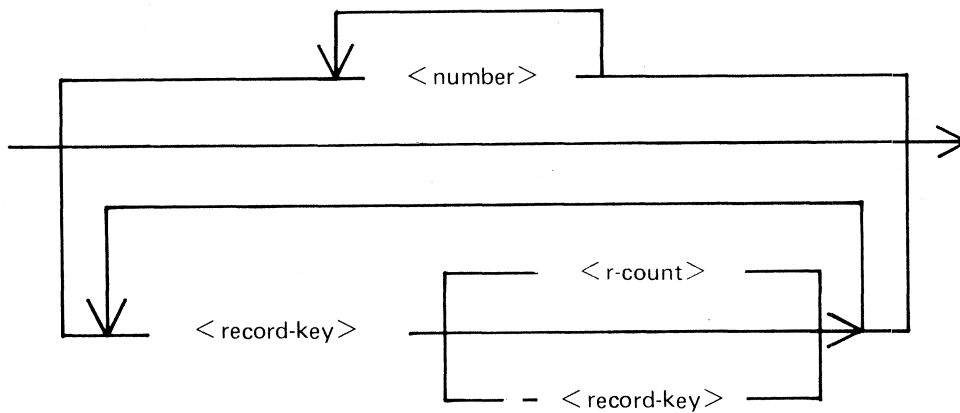


**COPY utility (Continued)**

< d-file > is defined as :



< r-spec > is defined as :

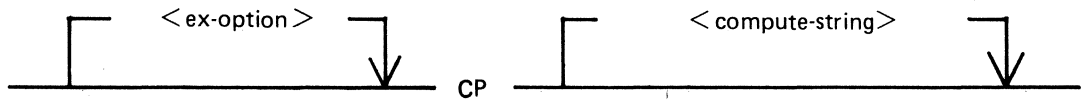




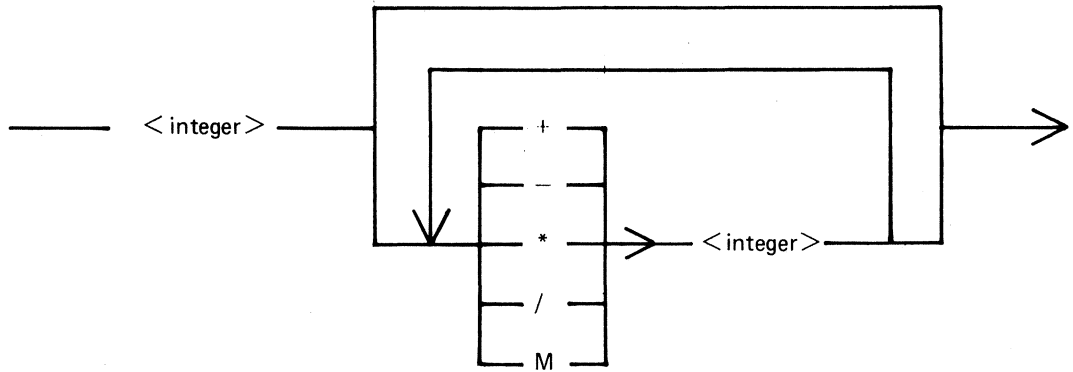
---

**CP utility**

---



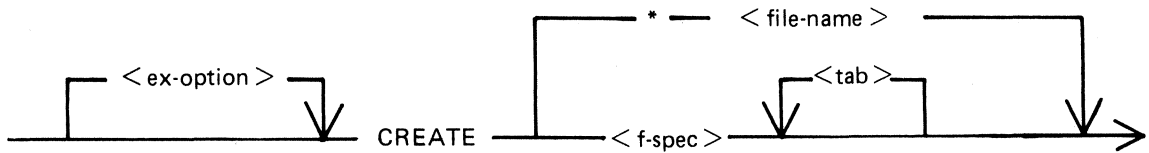
<compute-string> is defined as :



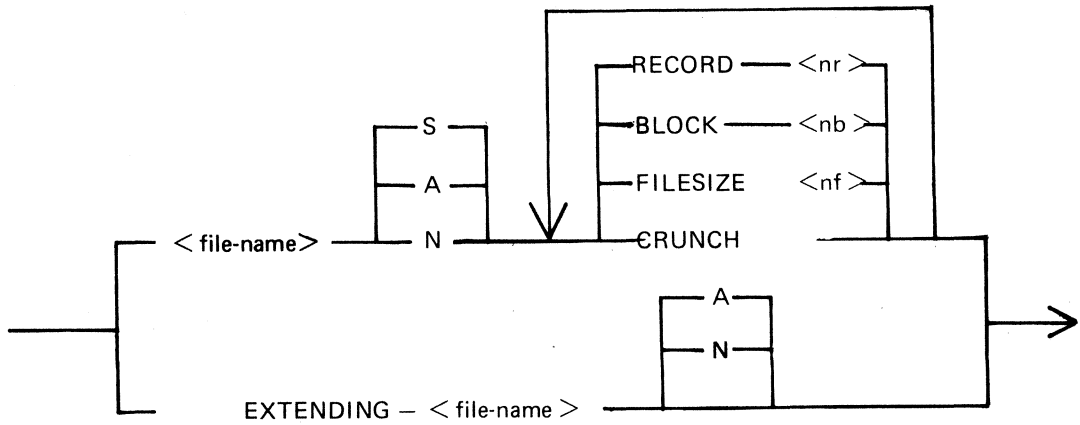

---

**CREATE utility**

---



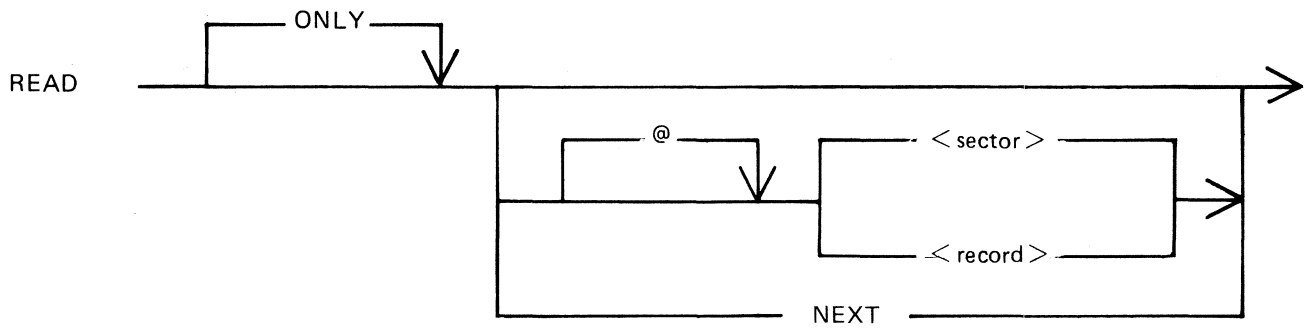
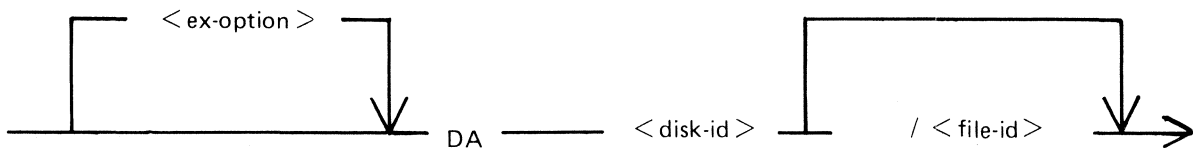
<f-spec> is defined as :



---

**DA utility**

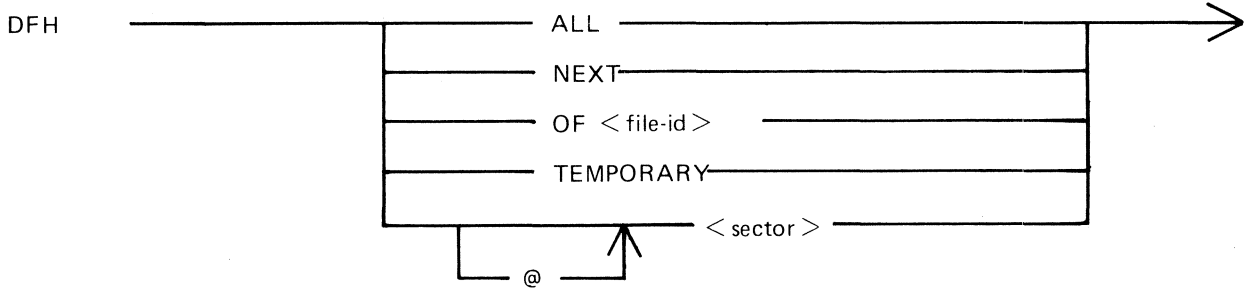
---



DISPLAY →

END →

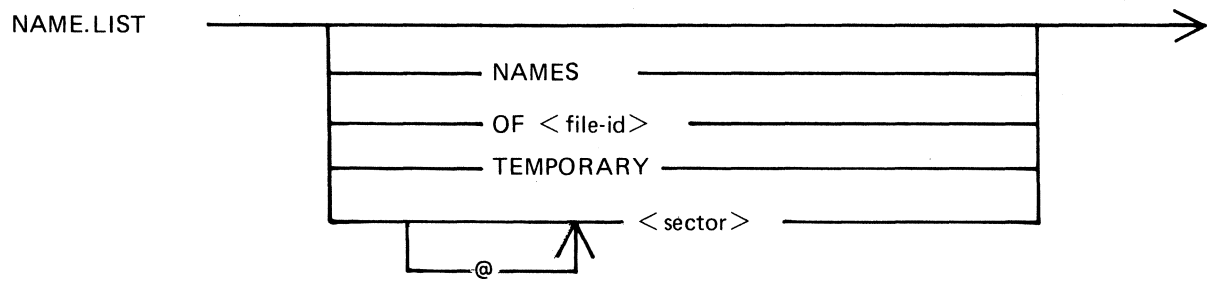
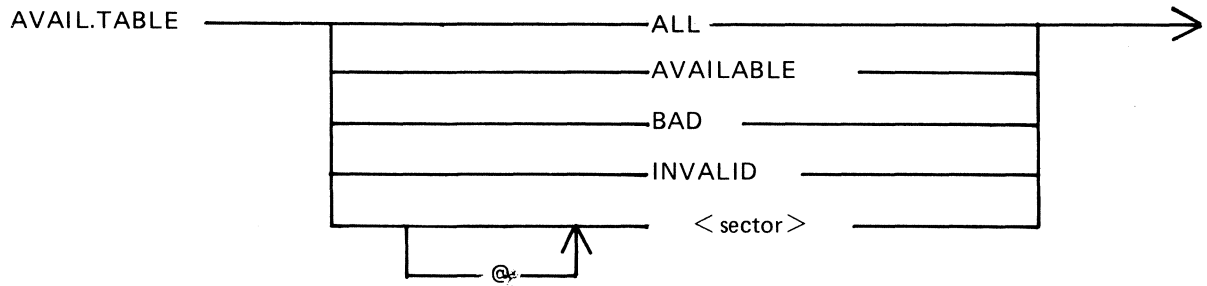
DCL →



---

**DA utility (Continued)**

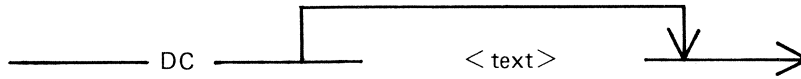
---



---

**DC intrinsic**

---



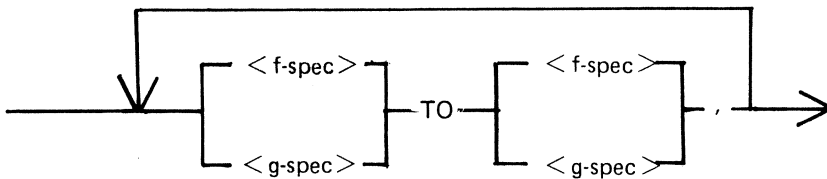
---

**DD utility**

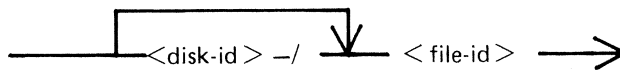
---

FUNCTION – STORE

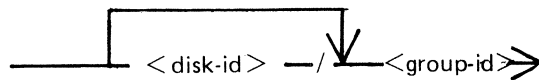
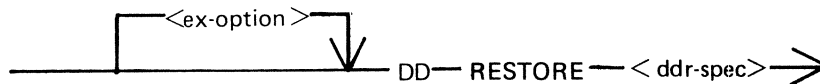
< dd-spec > is defined as :



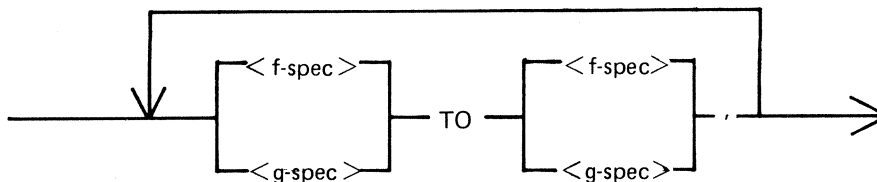
< f-spec > is defined as :



< g-spec > is defined as :

FUNCTION – RESTORE

< ddr-spec > is defined as :



The < f-spec > and < g-spec > are defined the same way as in STORE FUNCTION.

---

**DP intrinsic**

---

— DP — < mix no. > — / — < program-id > →

---

**DS intrinsic**

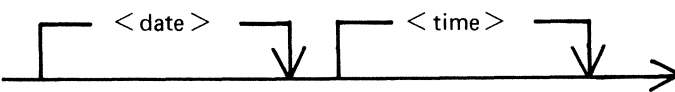
---

— DS — < mix > — / — < program-id > →

---

**DT intrinsic**

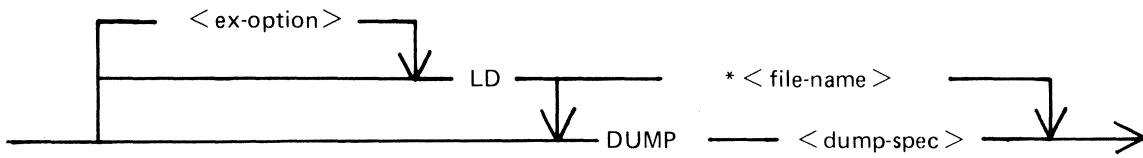
---

— DT — 

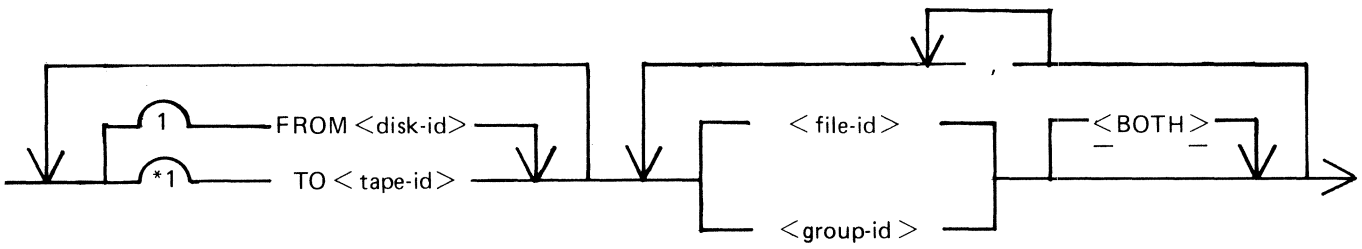
---

**DUMP utility**

---



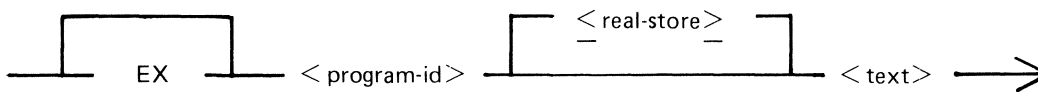
<dump-spec> is defined as :



---

**EX intrinsic**

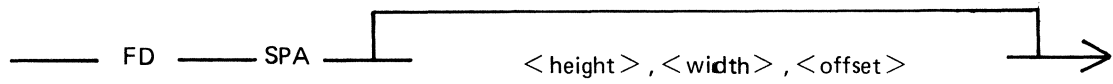
---



---

**FD intrinsic**

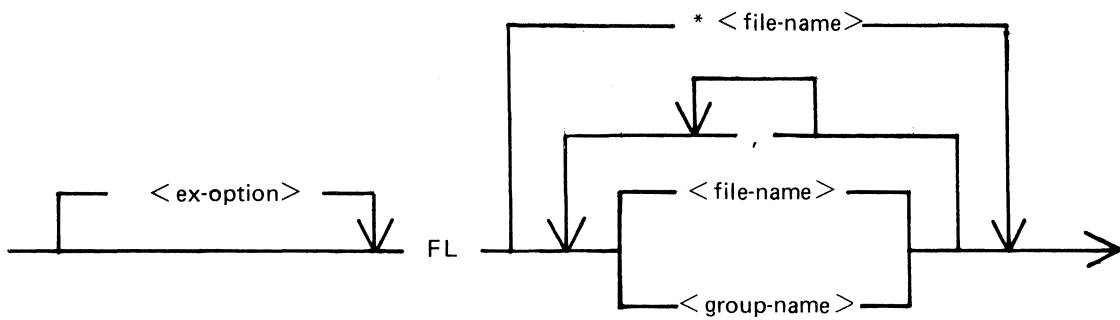
---



---

**FL utility**

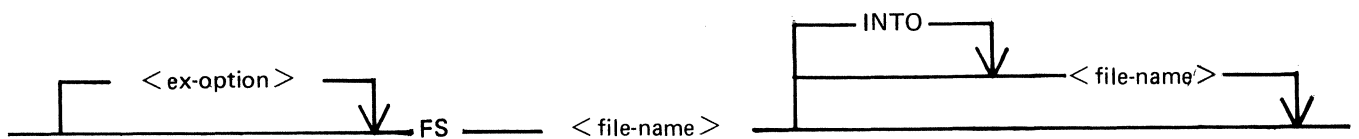
---



---

**FS utility**

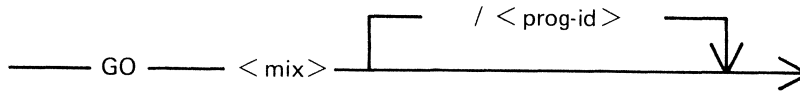
---



---

**GO intrinsic**

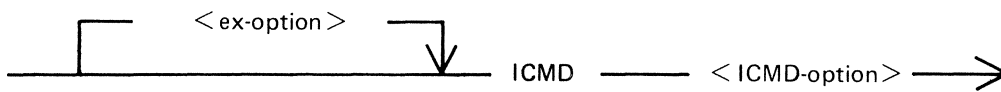
---



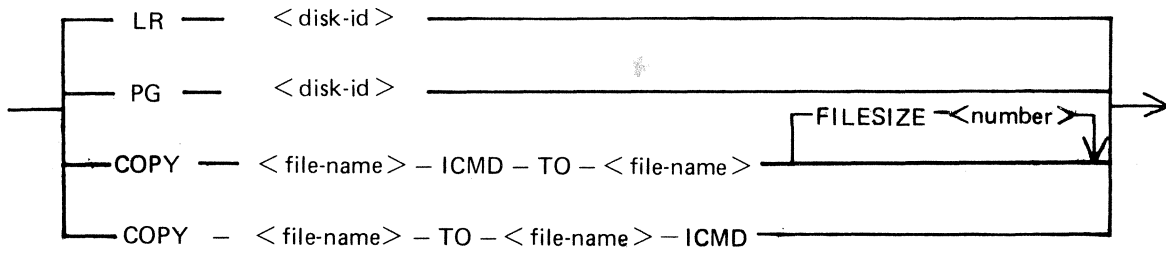
---

**ICMD utility**

---



< ICMD-option > is defined as :



< file-name > for this utility is defined as :

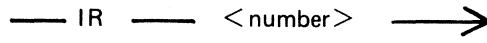
< disk-id > / < file-id >



---

**IR utility**

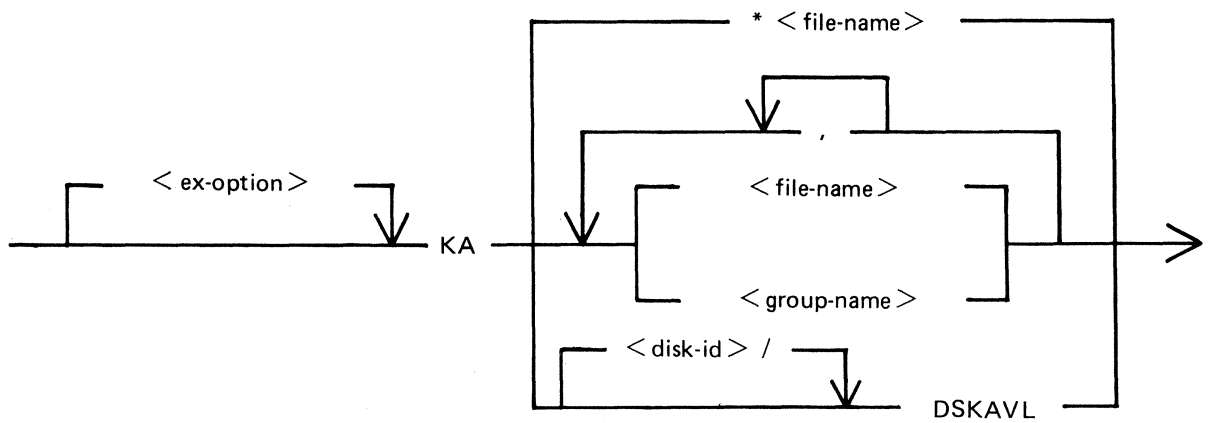
---



---

**KA utility**

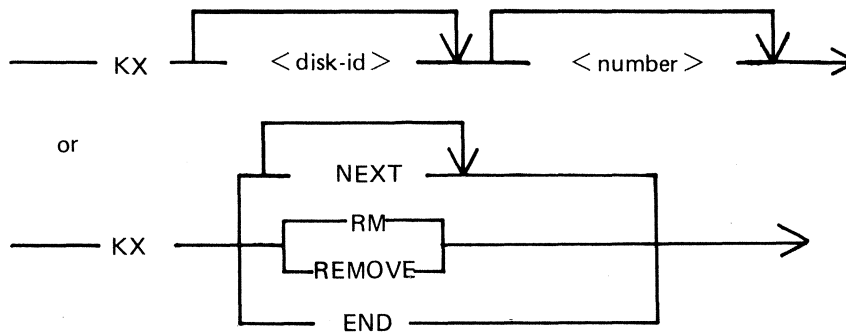
---



---

**KX utility**

---



---

**LB utility**

---

— LB —

---

**LD utility**

---

See ADD, DUMP, LOAD, and UNLOAD

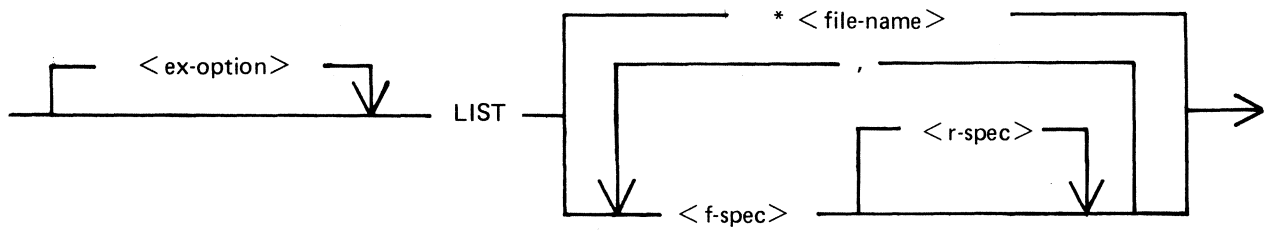
---

**LF utility**

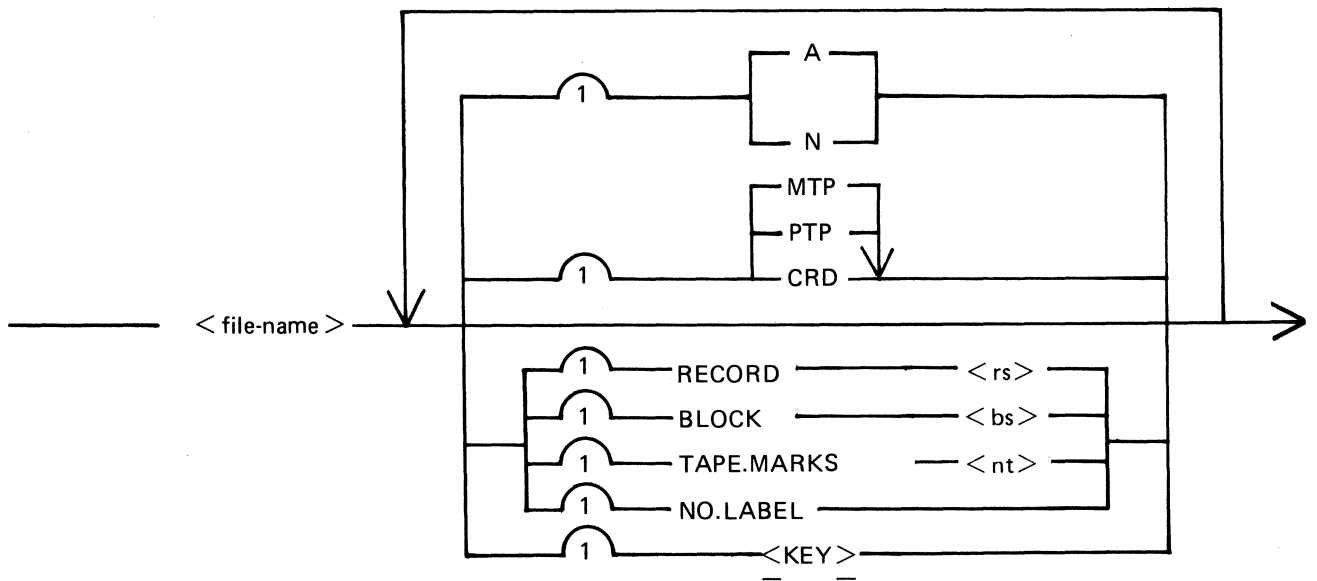
---

— LF —

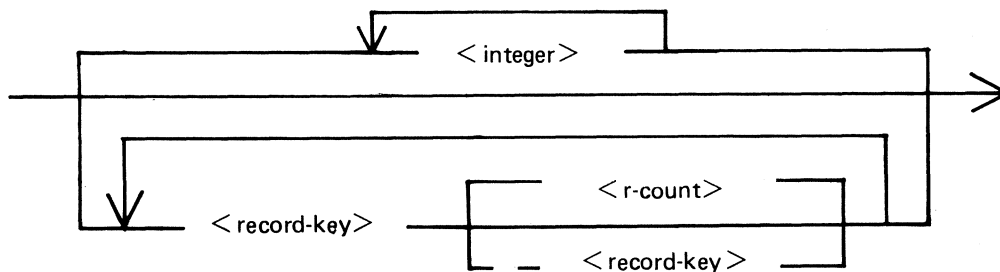
**LIST utility**



`<f-spec>` is defined as :



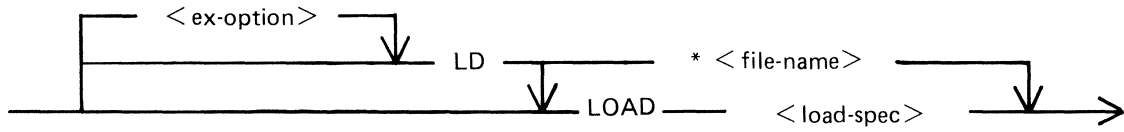
`<r-spec>` is defined as :



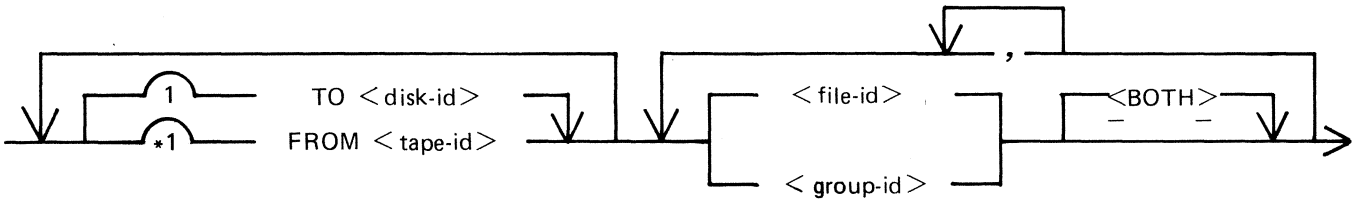
---

**LOAD utility**

---



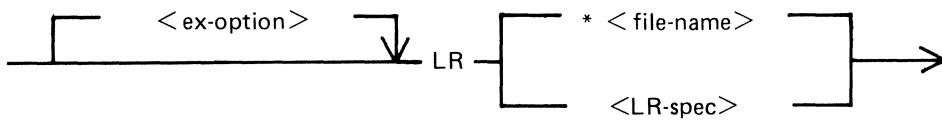
<load-spec> is defined as :



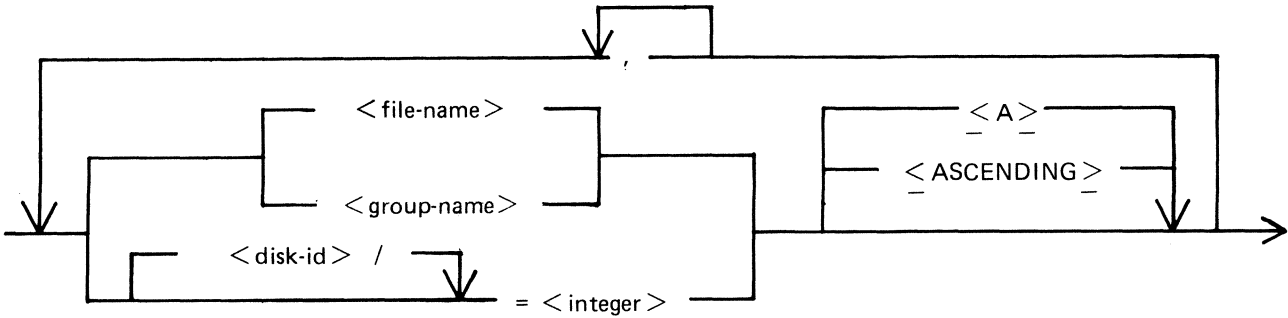
---

**LR utility**

---



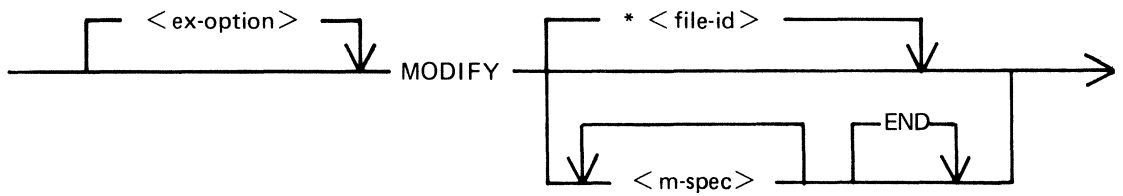
<LR-spec> is defined as :



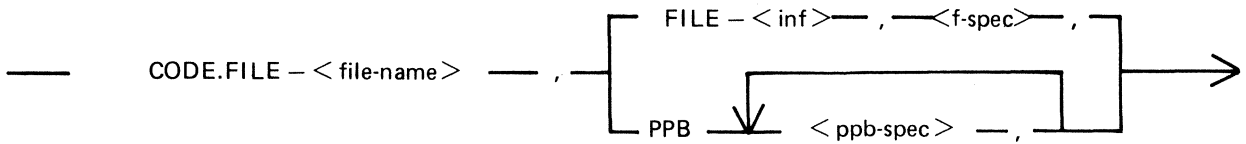
---

**MODIFY utility**

---



< m-spec > is defined as :



< f-spec > is defined as :



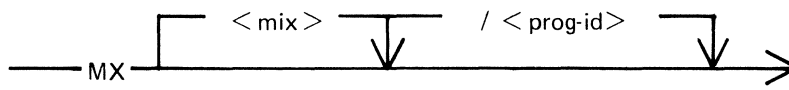
< ppb-spec > is defined as :



---

**MX intrinsic**

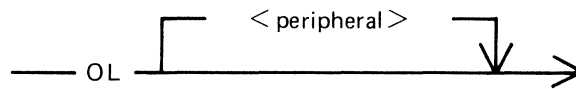
---



---

**OL intrinsic**

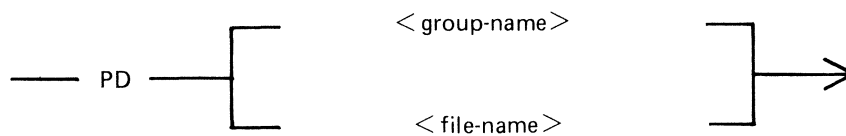
---



---

**PD utility**

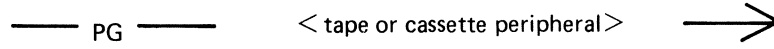
---



---

**PG intrinsic**

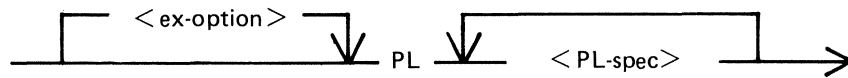
---



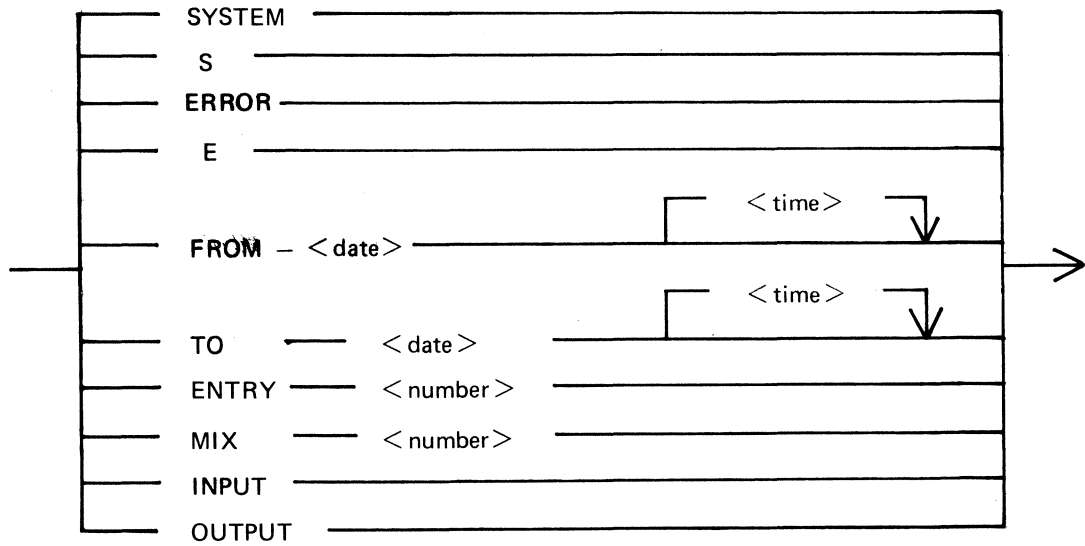
---

**PL utility**

---



< PL-spec > is defined as :



---

**PO intrinsic**

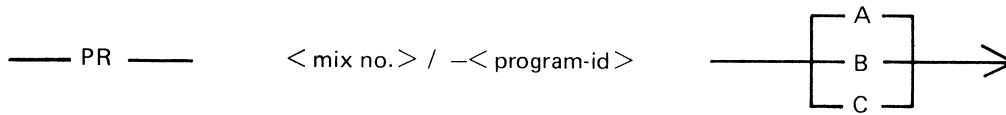
---

—— PO —— < peripheral > ——

---

**PR intrinsic**

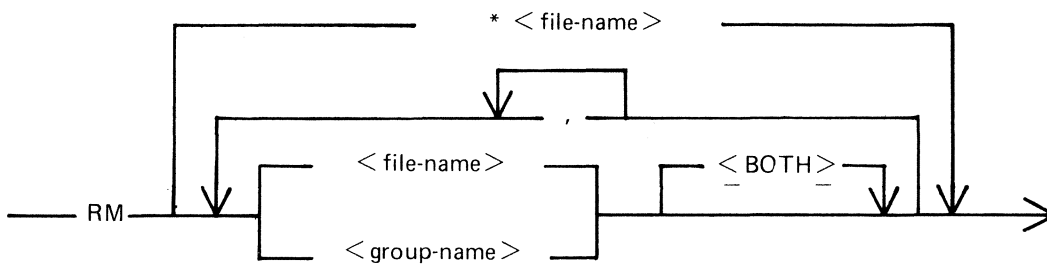
---



---

**RM utility**

---



---

**RY intrinsic**

---

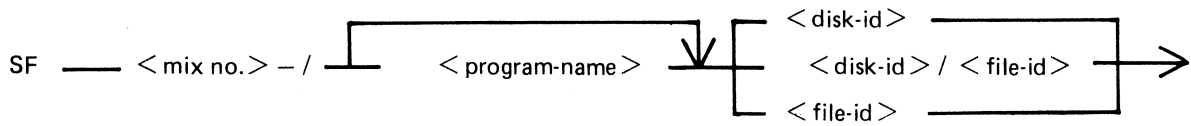
—— RY —— < peripheral > ——>



---

## SF intrinsic

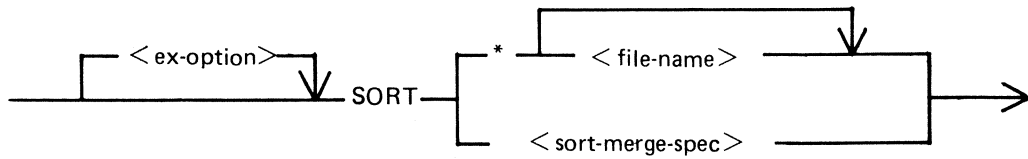
---



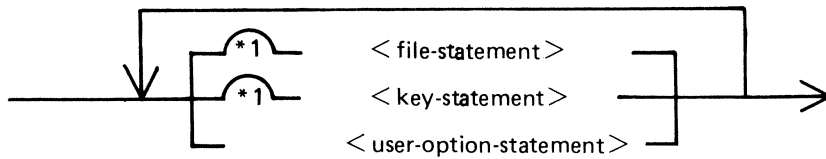

---

## SORT utility

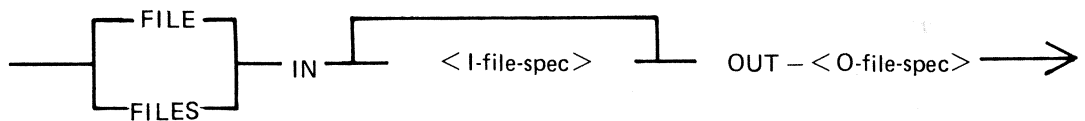
---



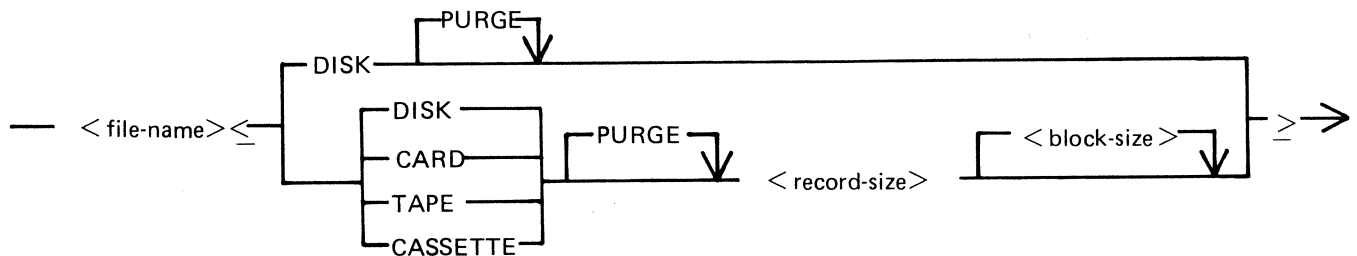
`<sort-merge-spec>` is defined as :



`<file-statement>` is defined as :

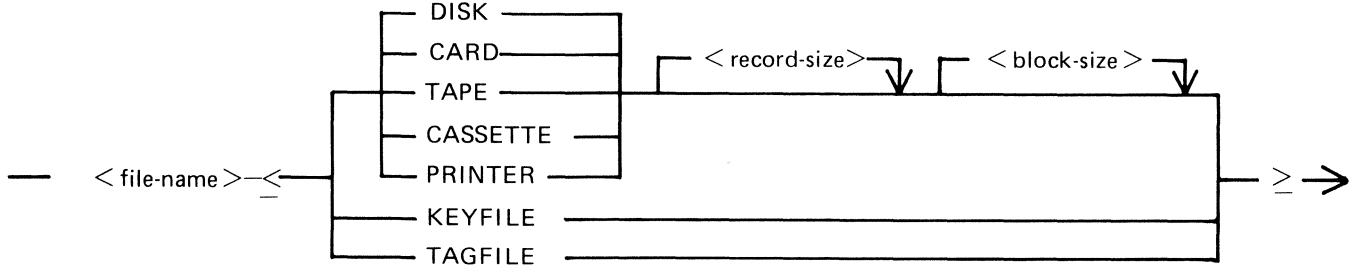


`<I-file-spec>` is defined as :

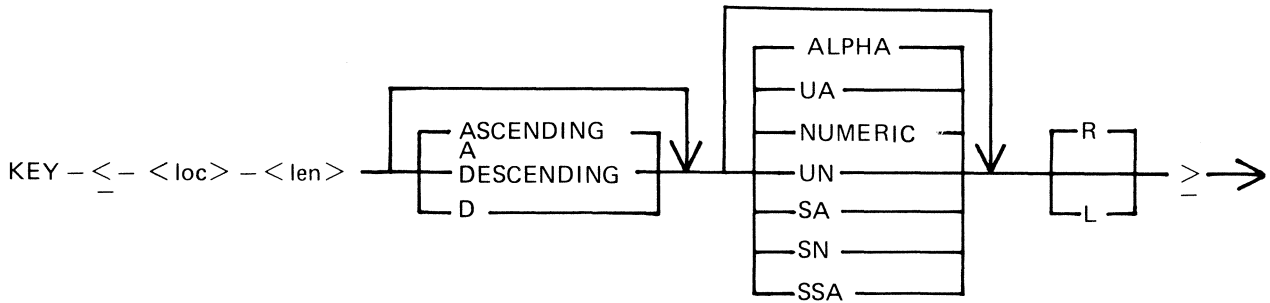


**SORT utility (Continued)**

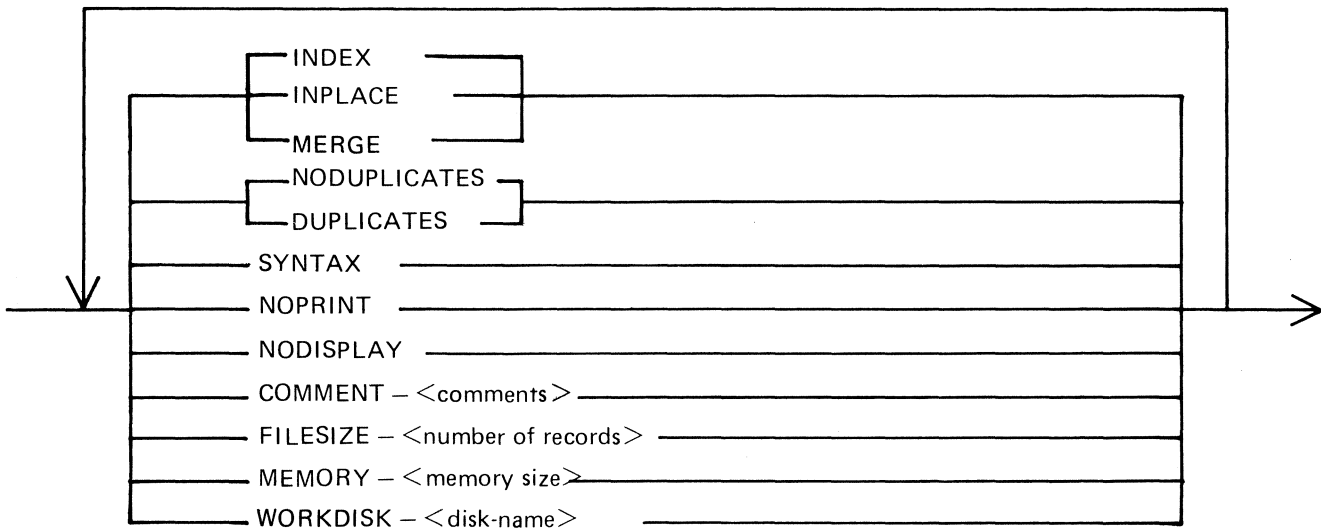
<O-file-spec> is defined as :



<key-statement> is defined as :



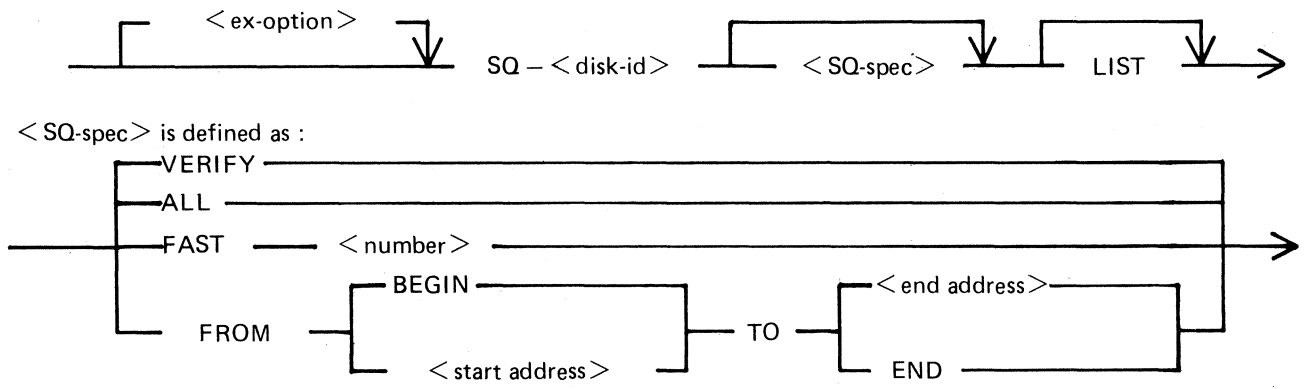
<user-option-statement> is defined as :



---

**SQ utility**

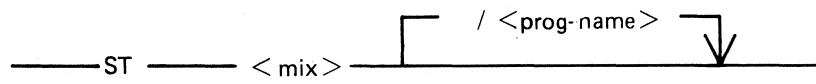
---



---

**ST intrinsic**

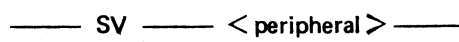
---



---

**SV intrinsic**

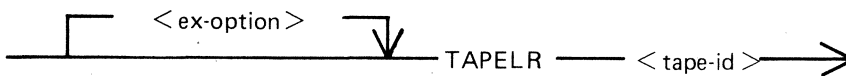
---



---

**TAPELR utility**

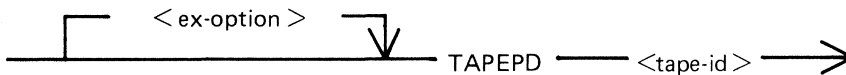
---



---

**TAPEPD utility**

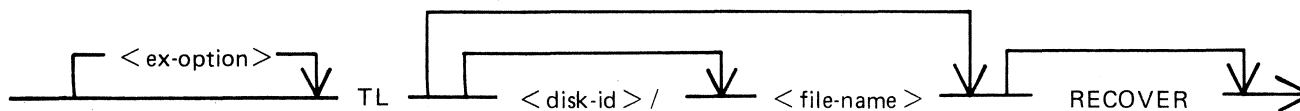
---



---

**TL utility**

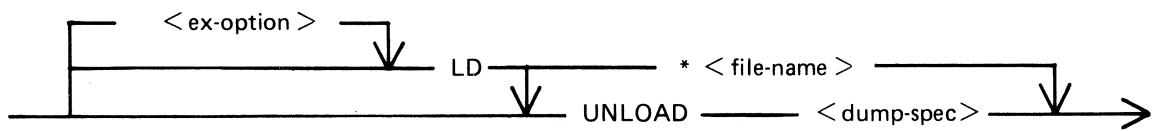
---



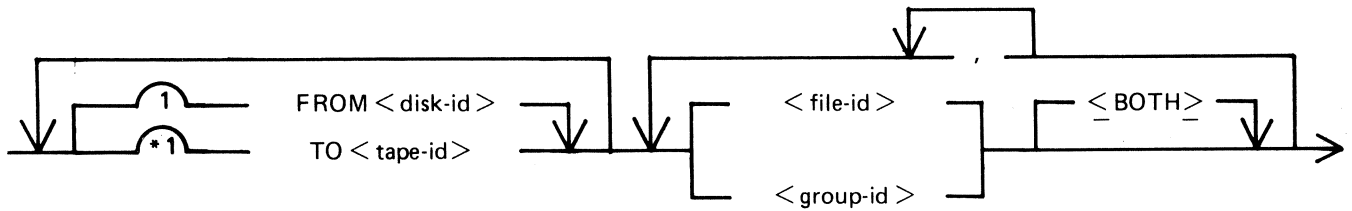
---

## UNLOAD utility

---



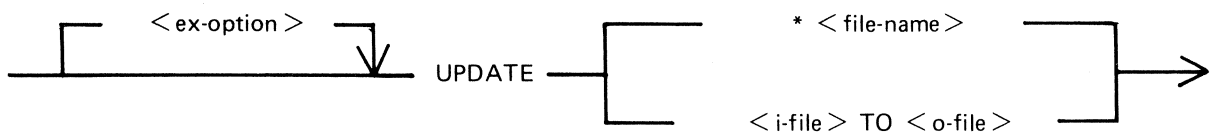
<dump-spec> is defined as :



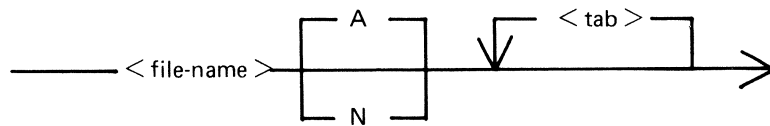

---

## UPDATE utility

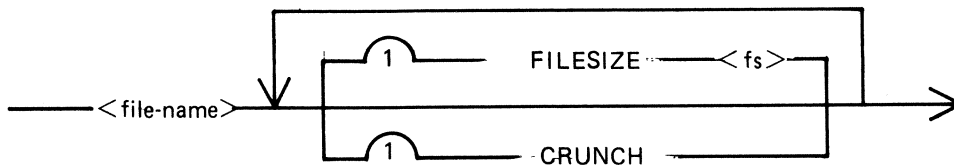
---



<i-file> is defined as :



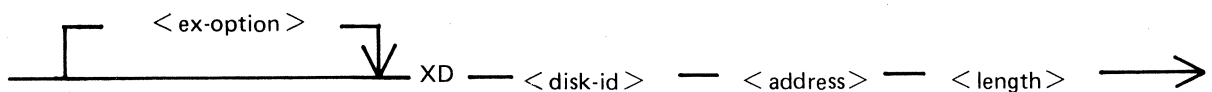
<o-file> is defined as :




---

## XD utility

---



# APPENDIX B

## EXAMPLES OF PRINTED UTILITY OUTPUT

This appendix provides sample output from some of the CMS—common utilities described in section 4.

SPO and console input messages are underlined. Some utilities use SPO display messages for output. Output print listings use a printer if one is available, or (for a B 80 or B 800) a console file. Print files can be either labelled or unlabelled : if a file is labelled, the name is printed following

?DATA

at the beginning of the listing, and

?END

at the end of the listing (for example, refer to the PL output listing). In this appendix, print files are shown boxed in : other output is on the SPO.

The meaning of the input messages are given in section 4. The utilities are given here in alphabetical order.

CHECKADUMP ARTAPE

NO DISCREPANCIES BETWEEN DUMP TAPE ART  
...APE AND DISK SYSTEM

CHECK.DISK MYDISK

10/CHECK.DISK < 2>MYDISK/SYSTEMEM DM READ  
... PARITY ERROR WHILE IN READ 227F  
ERROR NOTIFIED ON READING SECTOR 5489  
...  
10/CHECK.DISK < 2>MYDISK/SYSTEMEM DM READ  
... PARITY ERROR WHILE IN READ 227F  
ERROR NOTIFIED ON READING SECTOR 5552  
...  
10/CHECK.DISK < 2>MYDISK/SYSTEMEM DM READ  
... PARITY ERROR WHILE IN READ 227F  
ERROR NOTIFIED ON READING SECTOR 5553  
...  
CHECK.DISK ON MYDISK COMPLETED - ERRORS  
...NOTIFIED

COMPARE FRED WITH MYDISK/FRED  
FRED WITH MYDISK/FRED COMPARED - 1 ERRORS

DIFFERENCE DETECTED AT BYTE @002F@

FRED RECORD 2 IS:

@0000@ 5448 4953 2049 5320 4120 4455 4D4D 5920 4649 4C45 2043 5245 4154 4544 2054 4F20"THIS IS A DUMMY FILE CREATED TO "  
@0020@ 4445 4D4F 4E53 5452 4154 4520 484F 5778 2020 2020 2020 2020 2020 2020 2020 2020"DEMONSTRATE HOWx "  
@0040@ 2020 2020 2020 2020 2020 2020 2020 2020 " "

MYDISK/FRED RECORD 2 IS:

@0000@ 5448 4953 2049 5320 4120 4455 4D4D 5920 4649 4C45 2043 5245 4154 4544 2054 4F20"THIS IS A DUMMY FILE CREATED TO "  
@0020@ 4445 4D4F 4E53 5452 4154 4520 484F 5720 2020 2020 2020 2020 2020 2020 2020"DEMONSTRATE HOW "  
@0040@ 2020 2020 2020 2020 2020 2020 2020 2020 " "



DA MYDISK  
BOJ DA VERSION [3.01.01]

DATA CONS

£

DCL

CARTRIDGE IDENTIFIER OWNERS IDENTIFICATION	MYDISK TIO	SERIAL NUMBER	000009
INITIALIZATION DATE	79150	INITIALIZATION SYSTEM	BDS
PACK CODE	0	ACCESS CODE	
RESTRICTED CARTRIDGE	NO	INTEGRITY FLAG	0
BAD SECTOR COUNT	000000	ACTUAL ERROR COUNT	000000
NUMBER OF CYLINDERS	88/00580	UNIT OF ALLOCATION (SECTORS)	1/0010
NUMBER OF TRACKS/CYLINDER	2/0020	NUMBER OF SECTORS/TRACK	32/0020
NAME LIST ADDRESS	38/000260	NAME LIST LENGTH	5/0050
AVAILABLE TABLE ADDRESS	32/000200	AVAILABLE TABLE LENGTH	6/0060
ADDRESS OF FIRST DFH	43/000280	MAXIMUM NUMBER OF FILES	51/000330

£

DA (continued) :

DFH OF FRED

SECTOR: 44/@002C@ FILE IDENTIFIER FRED FILE TYPE @01@ = SOURCE LANGUAGE

CREATION DATE	79150	LAST ACCESS DATE	79150
GENERATION NUMBER	0/@0000@	IMPLEMENTATION LEVEL NUMBER	0/@00@
FLAGS	BIT 0 = 0	BIT 2 = 0	
RECORD SIZE	80/@0050@		
RECORDS/BLOCK	1/@0001@	SECTORS/BLOCK	1/@0001@
MAX FILE SIZE	54/@0036@	SAVE FACTOR -	(???)
MAX AREAS IN USE	1/@01@	OVERFLOW PACK-ID	???????
REC IN LAST AREA	4/@0004@	SPARE BYTES IN LAST RECORD	0/@0000@

USER COUNTS: TOTAL USERS - (0) BITS 0-2  
 OUTPUT USERS - (0) BIT 4  
 LOCK ACCESS USERS - (0) BITS 5-7

AREA	BIT MAP	AREA START ADDRESS	AREA SIZE	NUMBER OF RECORDS IN AREA
	1 2			
1	1 0 = ALLOCATED HERE	94/@005E@	54/@0036@	4/@0004@
2-16	0 0 = NOT ALLOCATED			

£

DA (continued):

DFH NEXT

SECTOR: 45/000200 FILE IDENTIFIER AMEND FILE TYPE 0100 = S-CODE

CREATION DATE 78107 LAST ACCESS DATE 79150  
 GENERATION NUMBER 0/000000 IMPLEMENTATION LEVEL NUMBER 0/0000  
 FLAGS BIT 0 = 0 BIT 2 = 0

RECORD SIZE 180/000B40  
 RECORDS/BLOCK 1/000010 SECTORS/BLOCK 1/000010  
 MAX FILE SIZE 36/000240 SAVE FACTOR - (???)

MAX AREAS IN USE 1/0010 OVERFLOW PACK-ID ???????  
 REC IN LAST AREA 36/000240 SPARE BYTES IN LAST RECORD 0/000000

USER COUNTS: TOTAL USERS - (0) BITS 0-2  
 OUTPUT USERS - (0) BIT 4  
 LOCK ACCESS USERS - (0) BITS 5-7

AREA	BIT MAP	AREA START ADDRESS	AREA SIZE	NUMBER OF RECORDS IN AREA
1	1 0 = ALLOCATED HERE	148/000940	36/000240	36/000240
2-16	0 0 = NOT ALLOCATED			

£

AVAIL. TABLE AVAILABLE

AVAILABLE TABLE ADDRESS: 32/000200 LENGTH: 6/000060

SECTOR	STATUS	LENGTH	START	END (+1)
32/000200	1 AVAIL	1038/0040E0	198/000C60	1236/004D40
	2 AVAIL	213/000D50	1913/007790	2126/0084E0
	3 AVAIL	259/001030	1336/005380	1595/0063B0
	4 AVAIL	354/001620	2216/008A80	2570/00A0A0
37/000250	29 AVAIL	3016/008C80	2616/00A380	5632/016000

£

DA (continued) :

NAME LIST NAMES

NAME LIST ADDRESS: 38/000260 LENGTH: 5/000050

SECTOR	STATUS	INDEX	DFH ADDRESS
38/000260	1 SYSEM	00	43/0002B0
	2 FRED	01	44/0002C0
	3 AMEND	02	45/0002D0
	4 XD	03	46/0002E0
39/000270	4 COBOL4	0E	57/000390
	8 COBOL1	12	61/0003D0
	9 COBOL5	13	62/0003E0
	10 COBOL3	14	63/0003F0
40/000280	1 COBOL7	16	65/000410
	6 COBOL	18	70/000460

£

READ 0040E

SECTOR	CHARACTER	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
0/000000	0251	008C	7150	4851	008D	8A4B	3738	5100	8D51	008C	4837	2F84	7189	5F84	718B	6F0C																			
	Q		P	K	Q		K	7	8	Q		Q		K	7	/																			
32/002000	2E40	0CB4	0D2E	400C	840D	9188	0C0D	0C51	008D	6102	D04F	0C42	0251	008C	994C	0C77																			
	. e		. e					Q			Q	B	Q		L																				
64/004000	0551	008D	8A4B	84F2	5F84	F36F	0C84	F65F	84F7	6F0C	0D84	FASF	84FB	6F0C	0D77	7484																			
	Q		K																																
96/006000	F65F	84F7	6F0C	771D	5100	8C71	501F	7763	AC77	0671	56E2	0802	0251	008C	5100	8D71																			
					Q		P	?			V		Q		Q																				
128/008000	505B	4837	4E84	FASF	84FB	6F0C	771D	5100	8C71	601F	773D	AC77	0671	57E2	0802	0251																			
	P	[	K	7	N			Q		?	=			W		Q																			
160/00A000	008C	5100	8D71	605B	4837	2851	008C	7150	1F77	2051																									
	Q			[	K	7	(	Q		P	?		Q																						

£

DA (continued) :

READ NEXT

SECTOR:	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0/e000e:	008C	7160	6F77	0871	62E2	0802	0237	1071	3CE2	0802	0251	008C	5100	8D71	605B	4818																
							7		<		Q		Q		[	K																
32/e020e:	2E40	0CB4	0D2E	400C	840D	9188	0C0D	0C51	008D	6102	D04F	0C42	0251	008C	994C	0C77																
	. e		. e					Q		D	B	Q			L																	
64/e040e:	0551	008D	8A4B	84F2	5F84	F36F	0C84	F65F	84F7	6F0C	0D84	FA5F	84FB	6F0C	0D77	7484																
	Q		K																													
96/e060e:	F65F	84F7	6F0C	771D	5100	8C71	501F	7763	AC77	0671	56E2	0802	0251	008C	5100	8D71																
					Q		P ?			V		Q		Q																		
128/e080e:	505B	4837	4E84	FA5F	84FB	6F0C	771D	5100	8C71	601F	773D	AC77	0671	57E2	0802	0251																
	P	[	K	7	N			Q		'	?	=		W		Q																
160/e0A0e:	008C	5100	8D71	605B	4837	2851	008C	7150	1F77	2051																						
	Q			[	K	7	(	Q		P	?	Q																				

£

END

EOJ DA

KA MYDISK DSKAM

WED 30 MAY 79		USAGE MAP OF MINI DISK MYDISK			SERIAL NO. 000009	OWNER TIO	PAGE 1.
INITIALISED 79150 ON		BDS FOR MAXIMUM OF 51 FILES			ALLOCATION UNIT 1	ERROR COUNT 000000	BAD SECTOR COUNT 000000
88 CYLINDERS		2 TRACKS PER CYLINDER		32 SECTORS PER TRACK			
AREA ADDRESS	AREA LENGTH	STATUS	FILE NAME				
94 @00005E@	1142 @000476@	AVAILABLE					
1336 @000538@	56 @000038@	AVAILABLE					
1522 @0005F2@	73 @000049@	AVAILABLE					
2216 @0008A8@	354 @000162@	AVAILABLE					
2620 @000A3C@	3012 @000BC4@	AVAILABLE					

KA MYDISK/=

WED 30 MAY 79		USAGE MAP OF MINI DISK MYDISK			SERIAL NO. 000009	OWNER TIO	PAGE 1.
INITIALISED 79150 ON		BDS FOR MAXIMUM OF 51 FILES			ALLOCATION UNIT 1	ERROR COUNT 000000	BAD SECTOR COUNT 000000
88 CYLINDERS		2 TRACKS PER CYLINDER		32 SECTORS PER TRACK			
AREA ADDRESS	AREA LENGTH	STATUS	FILE NAME				
0 @000000@	32 @000020@	ASSIGNED	*RESERVED				
32 @000020@	6 @000006@	ASSIGNED	*AVAIL.TABLE				
38 @000026@	5 @000005@	ASSIGNED	*FILE DIREC.				
43 @000028@	51 @000033@	ASSIGNED	*FILE HEADRS				
94 @00005E@	1142 @000476@	AVAILABLE					
1236 @0004D4@	100 @000064@	ASSIGNED	COBOL4				
1336 @000538@	56 @000038@	AVAILABLE					
1392 @000570@	130 @000082@	ASSIGNED	COBOL2				
1522 @0005F2@	73 @000049@	AVAILABLE					
1595 @000638@	128 @000080@	ASSIGNED	COBOL1				
1723 @000688@	91 @000058@	ASSIGNED	COBOL5				
1814 @000716@	99 @000063@	ASSIGNED	COBOL3				
1913 @000779@	213 @0000D5@	ASSIGNED	COBOL6				
2126 @00084E@	90 @00005A@	ASSIGNED	COBOL7				
2216 @0008A8@	354 @000162@	AVAILABLE					
2570 @000A0A@	46 @00002E@	ASSIGNED	COBOL				
2616 @000A38@	4 @000004@	ASSIGNED	FRED				
2620 @000A3C@	3012 @000BC4@	AVAILABLE					



LR MYDISK/=&lt;A&gt;

WED 30 MAY 79		DIRECTORY OF MINI DISK MYDISK						SERIAL NO. 000009	OWNER TIO	PAGE 1.	
INITIALISED 79150 ON		BDS FOR MAXIMUM OF		51 FILES		ALLOCATION UNIT		1	ERROR COUNT 000000	BAD SECTOR COUNT 000000	
88 CYLINDERS		2 TRACKS PER CYLINDER		32 SECTORS PER TRACK							
FILE NAME	ACTUAL SIZE	MAXIMUM SIZE	RECORD SIZE	RECS/ BLOCK	CREATED ACCESSED	FILE TYPE	NO. AREAS	AREA ADDRESSES	AREA SIZES	OVERFLOW DISK	
*RESERVED	32	32	180	32	79150 79150	SYSTEM	1	0 @000000@	32 @000020@		
*AVAIL.TABLE	6	6	180	32	79150 79150	SYSTEM	1	32 @000020@	6 @000006@		
*FILE DIREC.	5	5	180	32	79150 79150	SYSTEM	1	38 @000026@	5 @000005@		
*FILE HEADRS	51	51	180	32	79150 79150	SYSTEM	1	43 @000028@	51 @000033@		
AMEND	36	36	180	1	78107 79150	CODE 780226	1	148 @000094@	36 @000024@		
COBOL	46	46	180	1	79139 79150	CODE 790519	1	2570 @000A0A@	46 @00002E@		
COBOL1	128	128	180	1	78339 79150	CODE 780915	1	1595 @000638@	128 @000080@		
COBOL3	99	99	180	1	78339 79150	CODE 780525	1	1814 @000716@	99 @000063@		
COBOL4	100	100	180	1	78339 79150	CODE 780822	1	1236 @0004D4@	100 @000064@		
COBOL5	91	91	180	1	78339 79150	CODE 780830	1	1723 @000688@	91 @000058@		
COBOL7	90	90	180	1	78339 79150	CODE 780915	1	2126 @00084E@	90 @00005A@		
FRED	4	54	80	1	79150 79150	SRCELANG	1	94 @00005E@	54 @000036@		
XD	14	14	180	1	78107 79150	CODE 780226	1	184 @000088@	14 @00000E@		



LR MYDISK/= 100

WED 30 MAY 79		DIRECTORY OF MINI DISK MYDISK						SERIAL NO. 000009	OWNER TIO	PAGE 1.	
INITIALISED 79150 ON		BDS FOR MAXIMUM OF		51 FILES		ALLOCATION UNIT	1	ERROR COUNT 000000	BAD SECTOR COUNT 000000		
88 CYLINDERS		2 TRACKS PER CYLINDER		32 SECTORS PER TRACK							
FILE NAME	ACTUAL SIZE	MAXIMUM SIZE	RECORD SIZE	RECS/BLOCK	CREATED	FILE NO.	NO. AREAS	AREA ADDRESSES	AREA SIZES	OVERFLOW DISK	
COBOL1	128	128	180	1	78339 79150	CODE 780915	1	1595 @000638e	128 @000080e		
AREA ADDRESSES		AREA SIZES		AREA ADDRESSES		AREA SIZES		AREA ADDRESSES		AREA SIZES	
AVAILABLE AREAS											
-----											
198 @0000C6e	1038 @00040Ee	1913 @000779e	213 @0000D5e	1336 @000538e	259 @000103e						
2216 @0008A8e	354 @000162e	2616 @000A38e	3016 @0008C8e								
TOTAL AVAILABLE SPACE ON DISK				4880 @001310e							
AREA ADDRESSES		AREA SIZES		AREA ADDRESSES		AREA SIZES		AREA ADDRESSES		AREA SIZES	
TEMPORARY AREAS											
-----											
TOTAL SPACE ON DISK IN TEMPORARY USE				0 @000000e							

MODIFY

?DATA CON

CMS UTILITY: MODIFY EVERSION 3.01.013

USE PK1 FOR HELP

CODE.FILE? (PK1 depressed)

PK1=HELP PK2=MODIFY PPB PK3=MODIFY FPB PK4=CODE.FILE PK5=PRINT FPB/PPB PK6=TERMINATE

CODE.FILE? MYDISK/COBOL1

SELECT FUNCTION (PK2 depressed)

PPB ATTRIBUTE EOJ.SUPPRESS 0 (OFF) NEW VALUE ON

PPB ATTRIBUTE (PK5 depressed)

PPB OF CODE.FILE MYDISK /COBOL1

IMP.LEVEL.NO	0
PROGRAM NAME	"COBOL1 "
S-LANGUAGE	"BIL.REV.10 "
INTERP.PACK	"0000000"
INTERP.NAME	"BILINTERP "
COMPILER NAME	"BIL 3.0.2 "
COMPILE DATE	"780915"
EOJ.SUPPRESS	1
CLASS	4 (A)
INIT.MESS	@FF@
ENTRY POINT	0
PST.LENGTH	66
PST.LOCATION	2
DST.LENGTH	180
DST.LOCATION	3
TCB.PA LENGTH	88
TCB.PA LOCATION	92
STACK LENGTH	250
CCB.PA LENGTH	0
CCB.PA LOCATION	0
TCB.PE LENGTH	0
IFNB LENGTH	300
IFNB LOCATION	4
PPB ATTRIBUTE (PK6 depressed)	
...CHEERIO	

?END CON

MODIFY CODE.FILE MYDISK/COBOL1,PPB,EOJ.SUPPRESS OFF,PRINT.PPB,END

\* MODIFICATIONS SUCCESSFUL \*

?DATA LP

CMS UTILITY: MODIFY EVERSION 3.01.01J

CODE.FILE MYDISK/COBOL1,PPB,EOJ.SUPPRESS OFF,PRINT.PPB,END

PPB OF CODE.FILE MYDISK /COBOL1

IMP.LEVEL.NO		0
PROGRAM NAME	"COBOL1	"
S-LANGUAGE	"BIL.REV.10	"
INTERP.PACK	"0000000"	
INTERP.NAME	"BILINTERP	"
COMPILER NAME	"BIL 3.0.2	"
COMPILE DATE	"780915"	
EOJ.SUPPRESS		0
CLASS		4 (A)
INIT.MESS	@FF@	
ENTRY POINT		0
PST.LENGTH		66
PST.LOCATION		2
DST.LENGTH		180
DST.LOCATION		3
TCB.PA LENGTH		88
TCB.PA LOCATION		92
STACK LENGTH		250
CCB.PA LENGTH		0
CCB.PA LOCATION		0
TCB.PE LENGTH		0
IFNB LENGTH		300
IFNB LOCATION		4

?END LP

2015228

PL SYS-LOG-HOLD

?DATA LINES								
WED 6 JUN 79			CMS LOG FILE PRINTOUT OF SYS-LOG-HOLD				PAGE 1	
TIME HH:MM:SS	DATE MM/DD/YY	MESSAGE TYPE	MIX NUMBER	I/O MESSAGE	ENTRY NUMBER	RECORD NUMBER	MESSAGE	TEXT
	06/06/79	SYSTEM	15	INPUT	1	1	COMPARE FRED WITH MYDISK/FRED	
	06/06/79	SYSTEM	10	OUTPUT	2	1	END OF FILE FRED BEFORE MYDISK/F	
						2	RED - 0 ERRORS	
	06/06/79	SYSTEM	15	INPUT	3	1	COPY MYDISK/FRED TO FRED EXTENDI	
						2	NG	
	06/06/79	SYSTEM	10	OUTPUT	4	1	MYDISK/FRED TO FRED BAD ATTRIBUT	
						2	ES	
	06/06/79	SYSTEM	15	INPUT	5	1	COPY FRED TO FRED FILESIZE 10	
	06/06/79	SYSTEM	10	OUTPUT	6	1	FRED REMOVED	
	06/06/79	SYSTEM	10	OUTPUT	7	1	FRED TO FRED COPIED	
	06/06/79	SYSTEM	10	OUTPUT	8	1	NO RECORDS FOR COPYING FROM FRED	
	06/06/79	SYSTEM	15	INPUT	9	1	COPY MYDISK/FRED TO FRED EXTENDI	
						2	NG	
	06/06/79	SYSTEM	11	OUTPUT	10	1	MYDISK/FRED TO FRED BAD ATTRIBUT	
						2	ES	
	06/06/79	SYSTEM	15	INPUT	11	1	PO DMB	
	06/06/79	SYSTEM	15	OUTPUT	12	1	DMB O.K.	
	06/06/79	SYSTEM	9	OUTPUT	13	1	DMB E/ 0 FILES OPEN	
	06/06/79	SYSTEM	15	INPUT	14	1	PD E/=	
	06/06/79	SYSTEM	12	OUTPUT	15	1	E CONTAINS -	
	06/06/79	SYSTEM	12	OUTPUT	16	1	DELINP CMSCANDE DELINP.O	
	06/06/79	SYSTEM	12	OUTPUT	17	1	ORD.OLD.S NORD.O FACDAT	
	06/06/79	SYSTEM	12	OUTPUT	18	1	NORDO NORDS CUSMAS	
	06/06/79	SYSTEM	12	OUTPUT	19	1	END PD	

?END LINES

B-15

## SQ MYDISK FAST 2800 LIST

LARGEST AVAILABLE SPACE IS 3012 SECTORS

TOTAL AVAILABLE SPACE IS 4637 SECTORS IN

...5 AREA(S)

\*\*\* SQ COMPLETED \*\*\*

USAGE MAP OF DISK MYDISK BEFORE SQUASH

DATE : WED 30 MAY 79

PAGE 1.

FILE NAME	AREA FILE NO OPEN	DISK ALLOCATION LENGTH	FROM	TO	AVAILABLE TABLE INFORMATION	REMARKS
		1142	@00005E@	@0004D3@	AVAILABLE SPACE	
CD80L4	1	100	@0004D4@	@000537@		
		56	@000538@	@00056F@	AVAILABLE SPACE	
CD80L2	1	130	@000570@	@0005F1@		
		73	@0005F2@	@00063A@	AVAILABLE SPACE	
CD80L1	1	128	@00063B@	@00068A@		
CD80L5	1	91	@00068B@	@000715@		
CD80L3	1	99	@000716@	@000778@		
CD80L6	1	213	@000779@	@00084D@		
CD80L7	1	90	@00084E@	@0008A7@		
		354	@0008A8@	@000A09@	AVAILABLE SPACE	
CD80L	1	46	@000A0A@	@000A37@		
		1041	@000A38@	@000E48@	AVAILABLE SPACE	
FRED	1	4	@000E49@	@000E4C@		
		1971	@000E4D@	@0015FF@	AVAILABLE SPACE	

SQ (continued) :

LEASE MAP OF DISK MYDISK AFTER SQUASH				DATE : WED 30 MAY 79	PAGE 1.	
FILE NAME	AREA FILE NB OPEN	DISK ALLOCATION LENGTH	FROM	TO	AVAILABLE TABLE INFORMATION	REMARKS
		1142	@00005E@	@0004D3@	AVAILABLE SPACE	
COBOL4	1	100	@0004D4@	@000537@		
		56	@000538@	@00056F@	AVAILABLE SPACE	
COBOL2	1	130	@000570@	@0005F1@		
		73	@0005F2@	@00063A@	AVAILABLE SPACE	
COBOL1	1	128	@00063B@	@0006BA@		
COBOL5	1	91	@0006BB@	@000715@		
COBOL3	1	99	@000716@	@000778@		
COBOL6	1	213	@000779@	@00084D@		
COBOL7	1	90	@00084E@	@0008A7@		
		354	@0008A8@	@000A09@	AVAILABLE SPACE	
COBOL	1	46	@000A0A@	@000A37@		
FRED	1	4	@000A38@	@000A3B@		
		3012	@000A3C@	@0015FF@	AVAILABLE SPACE	

TAPELR ARTAPE

FILE NAME	ACTUAL SIZE	MAXIMUM SIZE	RECORD SIZE	RECS/ BLOCK	CREATED	ACCESSED	FILE TYPE
MYFILE	4	5	180	1	79172	79172	KEY
MYFILEQQ	6	10	50	1	79172	79172	DATA
A999	90	90	128	1	79158	79172	DATA

TAPEPD ARTAPE

NRZI TAPE ARTAPE <00000> DUMPED ON THU 21  
 ... JUN 79 CONTAINS -  
 MYFILE MYFILEQQ A999  
 END TAPEPD

# APPENDIX C

## GLOSSARY OF TECHNICAL TERMS

### **ADDRESS**

A disk is divided physically into tracks and sectors, both numbered sequentially from zero upwards. These 'numbers' are referred to as 'addresses'. The MCP uses this address scheme to quickly locate data on disk.

### **ALPHANUMERIC**

Consisting only of letters of the alphabet plus the ten numeric digits; that is, not containing any other special characters.

### **APPLICATION PROGRAM**

User program that performs day-to-day functions such as invoicing, printing, inventory reports, etc.

### **ATTRIBUTE**

Characteristic or quality.

### **BACK-UP**

Term used to describe the method of insuring that copies of files exist to standby as alternatives.

### **BINARY-CODED DECIMAL (BCD)**

A method of coding numeric information in 4-bit units representing 0 as bits 0000, 1 as bits 0001, 2 as bits 0010, up to 9 as bits 1001. For example, the number 1607 in BCD would take four 4-bit units (2 bytes), coded as 0001 0110 0000 0111.

### **BOJ**

'Beginning of Job' The term used to notify the operator that a program has entered the 'mix' and has just started running.

### **BSMD**

Abbreviation for 'Burroughs Super Mini Disk'.

### **BYTE**

One alphanumeric character of data.

### **CHECKERBOARDED**

Term applied to any disk having available spaces of varying sizes scattered about the disk amongst files. The term can also be applied to memory in a virtual memory system where 'locked' or 'save' areas are scattered through the memory in such a way as to impede getting overlayable memory areas of sufficient size for optimum throughput.



## **CMS**

Computer Management System. A set of interrelated specifications for system software, including high-level language compilers, object-code formats, operator interface and data communications, which Burroughs has implemented on machines of different hardware characteristics.

## **COMPILATION DATE**

The date on which a programmer's source code was compiled : that is, the creation date of the executable object program.

## **COMPILERS**

Group of system programs that convert instructions written by a programmer in a language such as COBOL or RPG into a form which can be run or interpreted by the hardware or system software.

## **CONFIGURATION**

Term used to describe the arrangement of various hardware devices in a particular system.

## **DATA FILE**

A set of information usually on a disk, which is used as data to be input.

## **DEFAULT VALUE**

Usually a meaning that a program will assume if not instructed otherwise.

## **DESTINATION**

Disk to which information is being transferred.

## **DISK DIRECTORY**

List, on Track 0, of file names, locations on disk, and sizes. Similar to a table of contents.

## **DISK FILE**

Set of information residing on a disk medium, collectively referred to by its name, 'file-name' and the name of the disk on which it resides ('disk-name').

## **DISK NAME**

Name by which a disk is known to MCP. Every disk medium has a 'label' of information written to it during disk initialization, and the disk name is part of the 'label'.

## **DUAL-PACK FILE (MULTI-VOLUME FILE)**

A file that resides on two separate disks or logically defined disks (for example, DKA, DKB).

## **EOJ**

'End of Job'. The term used to notify the operator that a program has terminated. 'Abnormal' end-of-job occurs when a program is terminated prematurely due to an error condition.

## **EXECUTION**

The running of a program is termed 'program execution'. The operator can execute (or start) a program by entering the name of the program desired (or disk-name/program if program resides on user disk). When a program is 'executed', it enters the 'mix' and is assigned a 'mix number' by the MCP.

## **FAMILY (GROUP) OF FILES**

Two or more disk files having at least the first letter of their names in common. For example, 'PR020', 'PRFILE', and 'PASM1' are members of a family of files that could be referred to as 'P—'.

## **HARDWARE**

Term referring to all equipment on the system. Line printers and disk cabinets are examples.

## **HEXADECIMAL ('HEX')**

A number system based on root 16, in contrast to common 'decimal' system based on root 10. To provide additional symbols, the letters A through F are used, so that counting proceeds thus: 0, 1, 2, 3, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12 ... for decimal numbers 0 through 18 ...

## **INTERPRETER**

A system software item used as an intermediate step in the running of a program. Instead of using a compiler to transform programs directly to machine instructions which can be run by the processor, a compiler may transform the program to an intermediate form (called 'S-code'). The S-code can be 'interpreted' by the interpreter, that is, translated into machine instructions that can be run by the processor.

## **INTRINSIC**

A 'command' used by the operator to direct the activities of the MCP. Intrinsic are actually a part of the MCP and therefore will never be seen on a disk file listing or in the 'mix'. Examples of intrinsic include 'DS' (discontinue the processing of a program) and 'DT' (retrieve or change system date).

## **KEYFILE**

File used by system as an index to a master data file.

## **LABEL**

A small space of disk on tape indicating the medium's contents, name, etc. A disk label may be created during the initialization process, and a tape label is created when the tape is purged.

## **MAIN MEMORY**

Circuit boards inside processor where program code and data in immediate use are held.

## **MCP ("Master Control Program")**

Program which is the central part of the CMS software system. It handles hardware devices, communicates with the operator, and controls processing of programs.

## **MIX**

Term applied to the mixture of programs running in a multi-programming environment. A 'mix-number' is a number which is assigned by the MCP to a program when it enters the 'mix'. A 'null' mix is when no jobs are running.

The program's name and mix number can be used by the operator to refer to a particular program in the 'mix'.

## **MULTIPROGRAMMING**

One processor working on more than one program at a time. Processing can be shared on a 'round-robin' basis, and computation can be overlapped with input/output if there is more than one program 'in the mix'.

## **ON-LINE**

Term used for equipment or media currently used as part of the system.

## **PACK**

Synonym for 'disk'

## **PERIPHERAL**

Hardware device used as input or output. Examples are line printer, disk drive unit, console keyboard.

## **PURGE**

To erase when disks or tapes are 'purged', their contents are lost.

## **SECTOR**

A disk is divided physically into data storage spaces called sectors, numbered sequentially from zero upwards. Each sector is 180 characters in length.

## **SOFTWARE**

Term referring to programs and files, as distinct from the 'hardware' of the actual machine.

## **SOURCE DISK**

Disk from which information is being transferred.

## **SOURCE FILE**

A disk file containing statements (instructions) written by a programmer in a high-level language such as COBOL or RPG, before it has been transformed into a runnable program.

## **STAND-ALONE PROGRAMS**

Programs that do not run under control of the MCP. In particular, functions of general use to all B80 users are held in a disk file called 'SAU'(Stand Alone Utilities). Examples include LS (list disk name and sizes), and RL (relabel a disk). Loading and execution of SAU is done with no need of the MCP. Refer to Section 8 for details.

## **STARFILE**

A small disk file optionally used at the start of most CMS-common utilities. The information in the starfile is used to build up the initiating message for the utility, which could also be entered by the operator on the SPO. Starfiles are also called 'macro-files'.

## **SYSTEM DISK**

The disk containing the copy of the MCP that is currently in use.

Note that a user disk may also contain MCP code files, but only the disk containing the MCP that is in use since the last warmstart is the system disk. There can be only one system disk at any time during operation. System disks cannot be used as system disks on more than one CMS product (see section 2 for details).

## **SYSTEM FILE**

A disk file which is used by the system software. Special control is placed on these files to minimise the danger of accidental removal from the disk (see RM utility).

## **SYSTEM SOFTWARE UTILITY**

A program of general use to all users, as opposed to an application program which performs a particular using day-to-day tasks, such as invoicing. Examples of utilities include COPY (copy files from one medium to another) and RM (remove files from a disk).

## **USER DISK**

Any disk available to the system that is not a system disk.

## **VIRTUAL MEMORY**

A software technique, implemented in the MCP, of allowing programs to execute (or several to execute together) when the total program memory requirements exceeds the amount of memory physically available. Some of the executing program's code and data, which is not in immediate use, is stored on disk media and not in main memory. When the code, or data, is required, space is made for it in main memory and the information read back from disk. To make space in memory, it may be necessary for the MCP to re-use some memory which has previously been used by the program and is not required at this moment. Before re-using memory containing data that could have been updated, the MCP writes this segment of memory to the program's 'virtual memory file' on disk.

This technique also applies to the code and data of the MCP and other system software.

## **VOLUME**

Synonym for 'disk'.

## **WRITE INHIBIT**

To prevent disk on tape media from being written to by a program. The manner in which this is accomplished depends upon the medium (see B80 or B800 System Operator manuals for details).

## **WRITE PERMIT**

To allow any disk or tape medium to be written to by a program. The manner in which this is accomplished depends upon the medium (see B 80 or B 800 System Operator manuals for details).

# APPENDIX D

## RELATED DOCUMENTATION

The following manuals provide information concerning CMS System Software:

<b>Manual</b>	<b>Form Number</b>
CMS ARCS Reference Manual	2012713
CMS COBOL Reference Manual	2007266
CMS MCP Reference Manual	2007266
CMS RPG Reference Manual	2007274
CMS MPLII Reference Manual	2007563
CMS NDL Reference Manual	1090925
CMS Data Communications Subsystem Reference Manual	1090909